PHP programozás

OOP 2. rész

Rostagni Csaba

2023. november 26.

Tartalom

Tartalom

Magic methods

- A PHP "mágikus metódusai" két darab aláhúzás jellel kezdődnek.

- `__construct()`
- `__destruct()`
- `__call()`
- `__callStatic()`
- `__get()`
- `__set()`
- `__isset()`
- `__unset()`
- `__sleep()`
- `__wakeup()`
- `__serialize()`
- `__unserialize()`
- `__toString()`
- `__invoke()`
- `__set_state()`
- `__clone()`
- `__debugInfo()`

Konstruktor

- A konstruktor PHP-ban az osztály neve helyett a `__construct()` magic method segítségével tudjuk létrehozni
- A régebbi verziókban (PHP3-4) a konstruktor neve megegyezett az osztály nevével
- Amennyiben névteret is használnunk csak a `__construct()` működik (PHP 5.5.3-tól)
- Öröklődés esetén nem hívja meg a szülő konstruktorát automatikusan

Link:

- Konstruktor és Dekonstruktor - php.net

Konstruktor

PHP

```php
class Diak1
{
    private $nev;
    private $kor;

    public function __construct($nev, $kor)
    {
        $this->nev = $nev;
        $this->kor = $kor;
    }
}
```

Konstruktor (típusokkal)

PHP

```php
class Diak2
{
    private string $nev;
    private int $kor;

    public function __construct(string $nev, int $kor)
    {
        $this->nev = $nev;
        $this->kor = $kor;
    }
}
```

Destruktor

- A destruktor `__destruct()` magic method lesz.
- A destruktor akkor hívodik rá, amikor már nem létezik ráhivatkozás, vagy a szkript futása véget ér (pl. `exit()` függvény hívásakor)
- Destruktorban nem dobunk kivételt!

Link:

- Konstruktor és Dekonstruktor - php.net

Destruktor példa

PHP

```php
class MyDestructableClass
{
    function __construct() {
        print "In constructor\n";
    }

    function __destruct() {
        print "Destroying " . __CLASS__ . ".\n";
    }
}

$obj = new MyDestructableClass();
exit();
```

Eredmény

```
Destroying MyDestructableClass.
```

Magic getter és setter

PHP

```php
class Auto {
  private string $gyarto;
  private string $tipus;
  private float $fogyasztas;

  public function __construct(string $gyarto,string $tipus,float $fogyasztas) {
      $this->gyarto = $gyarto;
      $this->tipus = $tipus;
      $this->fogyasztas = $fogyasztas;
  }

  public function __get($nev):mixed {
      return $this->$nev;
  }

  public function __set($nev,$ertek):void {
      $this->$nev = $ertek;
  }
}
```

Magic getter és setter használat közben

PHP

```php
$audi = new Auto("Audi", "A6",5.8);

echo "{$audi->gyarto} {$audi->tipus} ({$audi->fogyasztas})\n";

$audi->tipus = "A8";
$audi->fogyasztas = 6.1;

echo "{$audi->gyarto} {$audi->tipus} ({$audi->fogyasztas})\n";
```

Eredmény

```
Audi A6 (5.8)
Audi A8 (6.1)
```

- A tulajdonságok láthatósága privát
- Kiíráskor és értékadáskor hibát kellett volna dobnia
- Van `__get()` és `__set()` publikus metódusunk

Magic ___get magyarázat

PHP
```php
public function __get($nev):mixed {
    return $this->$nev;
}
```

PHP
```php
$audi->gyarto
```

- A `$audi->gyarto` meghívásakor a PHP meghívja a `__get()` metódust
- Az elérni kívánt privát tulajdonság a `gyarto`, így `$nev` változó értéke így `"gyarto"` lesz
- A `return $this->$nev` hívásakor a `$nev` helyére behelyettesítődik az értéke, így olyan, mintha a `return $this->gyarto`-t hívtuk volna meg

Magic ___set magyarázat

PHP
```php
public function __set($nev,$ertek):void {
    $this->$nev = $ertek;
}
```

PHP
```php
$audi->tipus = "A8";
```

- A `$audi->tipus = "A8"` meghívásakor a PHP meghívja a `__set()` metódust
- `$nev` változó értéke `"tipus"`, míg a `$ertek` változó értéke `"A8"` lesz
- A `$this->$nev = $ertek` hívásakor a `$nev` helyére behelyettesítődik az értéke, így olyan, mintha a `$this->tipus = "A8"`-at hívtuk volna meg

Tartalom

Scope Resolution Operator

- Röviden dupla kettőspont (`::`)
- Héberül: "Paamayim Nekudotayim"
- Osztályon kívül az osztály nevét kell megadni.
- Osztályon belül a `self`, `parent` és `static` kulcsszavakkal használatos.
- PHP 5.3.0 óta használható változókkal is, de a változó neve nem lehet a fenti 3 kulcsszavak egyike sem.

Link:

- `https://www.php.net/manual/en/language.oop5.paamayim-nekudotayim.php`
- `https://en.wiktionary.org/wiki/paamayim_nekudotayim`

Konstansok osztályon belül

- Létrehozhatunk osztály szinten létező konstansokat. (nem példány szintű!)
- Az értékük nem változhat.
- A konstans deklarálásánál kell megadni az értékét! Csak konstans kifejezés lehet, nem lehet változó, vagy metódus hívás.
- Az osztályon belül a `self` kulcsszóval érhetjük el (vagy az osztály nevével).
- Az osztályon kívül az osztály nevével érhetjük el.
- A `const` kulcsszóval vezetjük be, de a dollár jelet nem tesszük ki!
- Nagy betűvel illik írni.
- Amennyiben a láthatóság nincs megadva automatikusan `public` lesz. (kompatibilitási okokból)

Link:

- `https://www.php.net/manual/en/language.oop5.constants.php`

Konstansok használata

PHP

```php
class Kor{
    public const PI = 3.14;

    private $r;

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

    public function getTerulet()
    {
        return $this->r ** 2 * self::PI;
    }
}

$kor = new Kor(5);
echo "PI: " . Kor::PI . "\n";
echo "Terület:" . $kor->getTerulet();
```

Eredmény

```
PI: 3.14
Terület:  78.5
```

Osztály szintű metódusok

- A `static` kulcsszóval létrehozott metódusokból osztályonként egy közös létezik
- Az osztály példányosítása nélkül is használhatóak
- A `$this` kulcsszó nem használható benne
- Amennyiben a láthatóság nincs megadva, úgy automatikusan `public` lesz. (kompatibilitási okokból)
- Az UML osztálydiagram aláhúzással jelöli

Link:

- urlhttps://www.php.net/manual/en/language.oop5.static.php

Osztály szintű metódusok

PHP

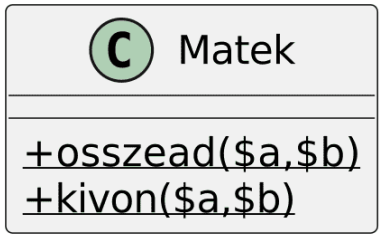
```php
class Matek {
    static function osszead($a,$b)
    {
        return $a + $b;
    }

    public static function kivon($a,$b)
    {
        return $a - $b;
    }
}

$x = 5;
$y = 10;

$z = Matek::osszead($x,$y);
echo $z . "\n";

echo Matek::kivon($z,8);
```



- Az `osszead()` publikus, pedig nem mondtuk meg neki.
- Az `osszead()` és a `kivon()` is meghívható a `Matek` példányosítása nélkül.

Eredmény

```
15
7
```

Osztály szintű tulajdonságok

- A `static` kulcsszóval létrehozott változókból és metódusokból osztályonként 1 létezik.
- Az osztály példányosítása nélkül is használhatóak.
- A `$this` kulcsszó nem használható benne.
- Amennyiben a láthatóság nincs megadva automatikusan `public` lesz. (kompatibilitási okokból)
- Az UML osztálydiagram aláhúzással jelöli

Link:

- urlhttps://www.php.net/manual/en/language.oop5.static.php

Osztály szintű tulajdonságok

PHP

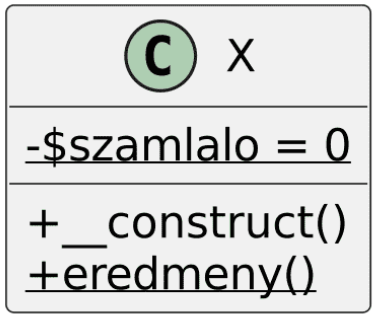
```php
class X {
    private static $szamlalo = 0;

    public function __construct()
    {
        self::$szamlalo++;
    }

    public static function eredmeny()
    {
        return self::$szamlalo;
    }
}

$x1 = new X();
$x2 = new X();
$x3 = new X();

echo X::eredmeny();
```



- A `self` kulcsszóval hivatkozhatunk rá.
- A `this` kulcsszó nem használható!
- A dollár jelet ki kell rakni az osztály szintű változóra hivatkozásnál!

Eredmény

```
3
```