PHP programozás

OOP 1. rész - alapok

Rostagni Csaba

2024. december 4.

Tartalom

Tartalom

PHP OOP Bevezető

- A PHP eredetileg nem objektum orientált nyelv.
- A PHP 5 verziótól kezdve használható.
- A PHP 5-ben meg lehetett adni a metódus paramétereinek az osztályát.
- A 7-es verziótól kezdve bővült a típusok listája, nem csak OOP-re korlátozódik.

Linkek:

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

Osztály

- Az osztály első eleme a `class` kulcsszó.
- Ezt követi az osztály neve.
- Végül a kapcsos zárójelek között az osztály tulajdonsági és metódusai.

PHP

```php
class OsztalyNeve
{
    // Tulajdonságok

    // Metódusok

}
```

Osztály

PHP

```php
class Tanulo
{
    private $nev;

    public function getNev()
    {
        return $this->nev;
    }

    public function setNev($nev)
    {
        $this->nev = $nev;
    }
}
```

Tartalom

Tulajdonságok (Properties)

- Az osztály tag változóit tulajdonságokat hivjuk.
  - Nem összekeverendő a C# Tulajdonságokkal!
- A láthatósága lehet `public`, `protected` vagy `private`
- Egy objektum tulajdonságait a `->` operátorral lehet elérni
- Inicializáláskor értéket is adhatunk, de csak konstans értéket

PHP
```php
class Osztaly
{
    public $a = 10;
}

$obj = new Osztaly;
echo $obj->a;
```

Tulajdnoságok deklarálása (NE így)

Mivel a PHP-ban nem kell előre deklarálni a változókat, így elviekben a következő kód működő képes lehet, de inkább kerüljük használatát.

Működik, de ezt NE!

```
class Y{ };
$y = new Y();
$y->a = 10;
echo $y->a;
```

Amennyiben hamarabb szeretnénk olvasni, mint írni a változóba figyelmeztetést kapunk

Rossz példa!

```
class X{ }
$x = new X();
echo $x->a;
```

Figyelmeztetés

```
Notice: Undefined property: X::$a
```

Tulajdonságok deklarálása

- A `public` láthatóságú tulajdonságok bárhol elérhetőek.
- A tulajdonságoknak deklaráláskor adhatunk alapértelmezett értéket.
- A következő kódok megsértik az OOP adatelrejtés alapelvét.

PHP

```php
class X
{
    public $a;
}

$x = new X();
$x->a = 10;
echo $x->a;
```

Eredmény

```
10
```

PHP

```php
class Y
{
    public $b = 10;
}

$y = new Y();

echo $y->b;
```

Eredmény

```
10
```

Tulajdonságok lekérése

A `private` tulajdonságok lekéréséhez külön (getter) metódus szükséges.

PHP

```php
class X
{
    private $a = 10;

    public function getA() {
        return $this->a;
    }
    // ...
}

$x = new X();
echo $x->getA();    // 10
```

Tulajdonságok módosítása

A `private` tulajdonságok módosításához külön (setter) metódus szükséges.

PHP

```php
class X
{
    private $a = 10;
    // ...
    public function setA($a){
        $this->a = $a;
    }
}

$x = new X();
$x->setA(8);
echo $x->getA();    // 8
```

Tartalom

Metódusok

- Adjuk meg a láthatóságot: `public`, `protected` vagy `private`
- Amennyiben elhagynánk, akkor `public` lesz az alpértelmezett
- A metódusokat is a `function` kulcsszóval adjuk meg
- A metódus nevét kerek zárójel pár követi

PHP

```php
class X
{
    public function hello($nev)
    {
        return "Hello {$nev}!";
    }
}
$x = new X();
echo $x->hello("Bence");
```

Eredmény

```
Hello Bence!
```

Tartalom

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;
    }
}
```

Tartalom

Első osztályunk

Matek1.php

```php
class Matek1 {
    private int $a;
    private int $b;

    public function __construct(int $a, int $b) {
        $this->a = $a;
        $this->b = $b;
    }

    function osszead()  {
        return $this->a + $this->b;
    }
}
```

Mappaszerkezet:

```
/
├── index.php
└── Matek1.php
```

index.php

```php
include('./Matek1.php');

$m1 =  new Matek1(8,12);
echo $m1->osszead();  // 20
```

Első osztályunk

- Előkészítés
  - Az osztály neve kezdődjön nagy betűvel!
  - Minden ostzály kerüljön külön fájlba!
  - A fájl és az osztály neve legyen ugyanaz!
  - A kis- és nagybetűket tükrözze hűen a fájl elnevezése!
- Használat
  - Az osztályunkat tartalmazó fájlt bele kell foglalni abba a szkriptbe, amiben szetetnénk használni. (a példában `include()`)

Tartalom

Névterek

- Előfordulhat, hogy ugyanazt az osztálynevet több osztályunk használná.
- Több külsős kód használatánál összeakadások lehetnek.
- Megoldás: névtér.
- Fontos, hogy a névtér megadása legyen az első utasítás a fájlban!

PHP
```php
namespace nev;
```

http://nyelvek.inf.elte.hu/leirasok/PHP/index.php?chapter=6

Globális névtér

- A \ jellel hivatkozhatunk rá.
- A beépített függvények és osztályok a globális névtérben találhatóak.
- Csak akkor szükséges használni, ha meghatároztuk a névteret.

A DateTime egy beépített osztály, alap esetben használhatjuk gond nélkül:

PHP
```php
$datumido = new DateTime()
```

Amikor megadtuk a fájlunkban, hogy melyik névtérben vagyunk, ott meg kell adni, hogy a globális névtérben lévő DateTime-ot szeretnénk használni, különben a mi névterünkben keresné a DateTime osztályt.

PHP
```php
namespace nevterem;

$datumido = new \DateTime();
```

Globális névtér használata

Rossz példa!

```php
<?php
namespace nevterem;
$datumido = new DateTime();
echo $datumido->format("H-i-s");
```

Eredmény

```
Fatal error: Uncaught Error: Class "nevterem\DateTime"
not found in ... thrown in ... test.php on line 3
```

PHP

```php
<?php
namespace nevterem;
$datumido = new \DateTime();
echo $datumido->format("H-i-s");
```

Eredmény

```
09-36-15
```

Teljesen minősített osztály név (PSR-4)

`\<NamespaceName>(\<SubNamespaceNames>)*\<ClassName>`

- Rendelkeznie **kell** felső szintű névtérrel. (pl.: Acme)
- **Rendelkezhet** egy vagy több alnévtérrel. (pl.: Matek és Geometria)
- A legvégén az osztály nevének **kell** szerepelnie. (pl.: Kor)

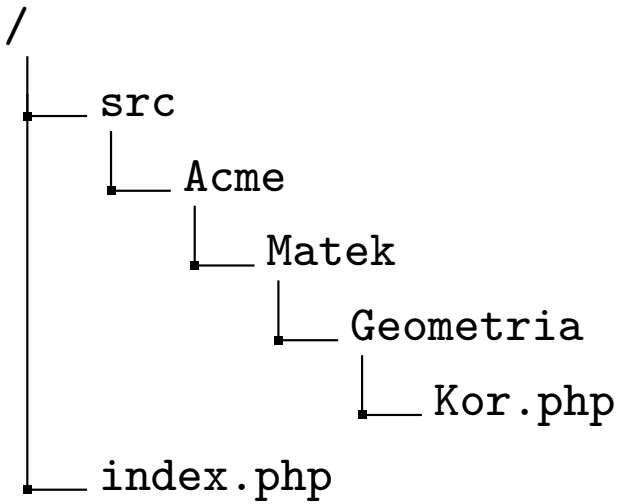
például:

`\Acme\Matek\Geometria\Kor`

Linkek:

- `https://www.php-fig.org/psr/psr-4/`
- `https://github.com/php-fig/fig-standards/blob/master/accepted/PSR-4-autoloader-examples.md`

Osztályok és névterek



Kor.php

```php
<?php
namespace Acme\Matek\Geometria;

class Kor
{
    ...
}
```

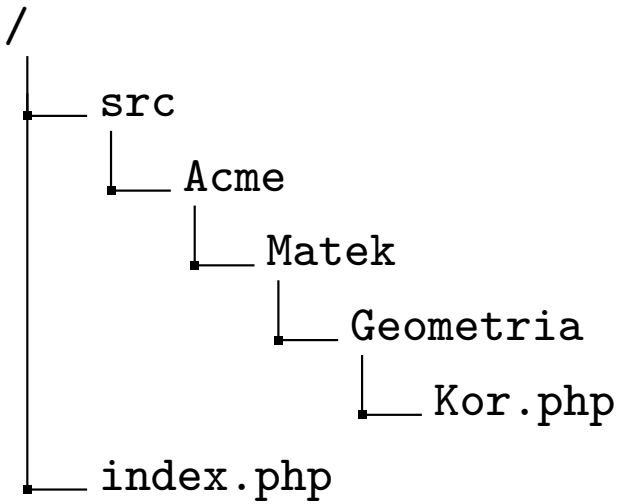
PHP

```php
<?php
include('./src/Acme/Matek/Geometria/Kor.php');

$kor1 = new \Acme\Matek\Geometria\Kor(5);
$kor2 = new \Acme\Matek\Geometria\Kor(8);
```

- Elég hosszadalmas minden esetben megadni az osztály teljes nevét.

Osztályok és névterek



Kor.php

```php
<?php
namespace Acme\Matek\Geometria;

class Kor
{
    ...
}
```

PHP

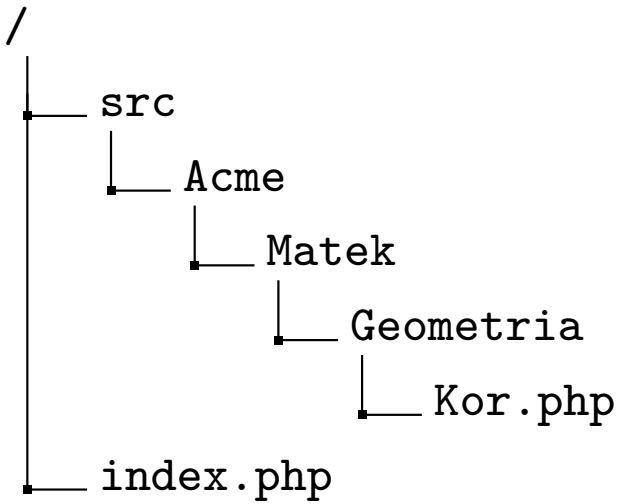
```php
<?php
include('./src/Acme/Matek/Geometria/Kor.php');

use \Acme\Matek\Geometria\Kor;

$kor1 = new Kor(5);
$kor2 = new Kor(8);
```

- A `use` operátor segítségével leegyszerűsíthetjük a kódot.

Osztályok és névterek



Kor.php

```php
<?php
namespace Acme\Matek\Geometria;

class Kor
{
    ...
}
```

PHP

```php
<?php
include('./src/Acme/Matek/Geometria/Kor.php');

use \Acme\Matek\Geometria\Kor as Karika;

$kor1 = new Karika(5);
$kor2 = new Karika(8);
```

- A `use` operátor segítségével aliast is létrehozhatunk.

Osztályok és névterek

Kor.php (Kutya)

```php
<?php
namespace Acme\Menhely\Kutya;

class Kor
{
    ...
}
```

Kor.php (Geomertia)

```php
<?php
namespace Acme\Matek\Geometria;

class Kor
{
    ...
}
```

PHP

```php
<?php
include('./src/Acme/Matek/Geometria/Kor.php');
include('./src/Acme/Menhely/Kutya/Kor.php');

use \Acme\Matek\Geometria\Kor as Karika;
use \Acme\Menhely\Kutya\Kor as KutyaKor;

$matekosKor = new Karika(5);
$kutyaKora = new KutyaKor(8);
```