Dr. Pál László, Sapientia EMTE, Csíkszereda

WEB PROGRAMOZÁS
2.ELŐADÁS

2015-2016

Objektumorientált programozás

OOP PHP-ben

- A PHP az 5.0-as verziójától megvalósítja az OO eszközrendszerét
- OO eszközök:
  - Osztályok
  - Objektumok
  - Öröklés
  - Interfészek

OOP PHP-ben

- Miért fontos ismerni?
  - Bonyolult függvények használata nehezíti a programírást
  - A CMS rendszerek (Joomla, Drupal, Wordpress, stb.), egyéb keretrendszerek (CodeIgniter, Symfoni, Zend, stb.) ezen módszerrel vannak megvalósítva
- Az OOP a PHP-ben hasonló felépítéssel rendelkezik, mint más OOP alapú nyelvekben (pld. Java, C#)

Osztály

- Az osztályokat a *class* kulcsszóval kezdjük, ezt követi az osztály neve, majd a két kapcsos zárójel, amelyek között szerepelnek a tulajdonságok és a hozzájuk tartozó metódusok
- Szintaxis:

```
class osztlay{
    adattagok
    metodusok
}
```

```php
class EgyszeruOsztaly {

// tulajdonságdeklaráció
    public $valtozo = 'alapértelmezett érték';

// metódus deklaráció
    public function mutatValtozo() {
        echo $this->valtozo;
    }
}
```

Osztály

- Példák osztálydeklarációkra:

```php
class Kocsi
{
    public $szin;
    public $gyartmany;
    public $ar;
}
```

```php
class Szemely
{
    public $nev;
    public $kor;
    public $lakcim;
}
```

```php
class Aru
{
    public $kod;
    public $nev;
    public $ar;
}
```

Osztály példányosítása

- Példa:

```
class osztalyom {

    public $attributum;

    public function metodus() {
        echo $this->attributum;
    }
}

$peldany = new osztalyom();
$peldany->attributum = 'Hello World';
$peldany->metodus();
?>
```

- Új példány:

```
$peldany = new
        osztalyom();
```

- Az objektumok tagjait a

-> operátorral lehet elérni

- A tagfüggvényeken belül az aktuális objektumra a **$this** speciális változóval hivatkozhatunk

Osztály példányosítása – Példa


```php
class User {

    public $vezNev;
    public $kerNev;
    public $szulEv;

    function teljesNev() {
        return $this->vezNev . " " . $this->kerNev;
    }

    function kor() {
        return date("Y") - $this->szulEv;
    }

    function koszont() {
        print("Szia " . $this->teljesNev() . "!");
    }
}
```

```php
$pityu = new User();
$pityu->kerNev = "Kiss";
$pityu->vezNev = "Istvan";
$pityu->szulEv = 1980;
$pityu->koszont();
```

Tagváltozók láthatósága

- public $vezNev;
  - Elérhető kívülről, az objektumváltozón keresztül
    - Kívülről: $pityu->vezNev
    - Tagfüggvényből: $this->vezNev
- private $szamlaszam;
  - Csak az objektum tagfüggvényeiből érhető el
    - Kívülről: $jani->szamlaszam HIBA!
    - Tagfüggvényből: $this->szamlaszam

Konstruktorok

- Konstruktor: egy speciális metódus, amely egy objektum példányosítása során automatikusan meghívódik, akkor is ha azt nem definiáltuk
  - Fogadhat paramétereket is: ezeket a new parancsnál az osztály neve után kell megadni
- Példa:

```
function __construct($vez) {
    $this->vezNev = $vez;
}
```

```
$feri = new User("Puskas");
```

Konstruktorok – Példa


```php
class Auto {

    private $szin;
    private $gyarto;
    private $motorszam;

    public function __construct($gyarto, $motorszam, $szin) {
        $this->gyarto = $gyarto;
        $this->motorszam = $motorszam;
        $this->szin = $szin;
    }
```

```php
function irdkiAutoJellemzo() {
    echo "Gyarto:" . $this->gyarto . "<br />";
    echo "Motor szám:" . $this->motorszam. "<br />";
    echo "Szín:" . $this->szin. "<br />";
}
```

```php
$audi = new Auto('Audi', 123, 'Piros');
$audi->irdkiAutoJellemzo();
```

Destruktorok

- Objektumok megsemmisülése után hívódik meg (ha nem létezik több referencia az objektumra)
- Szintaxis:

```
public function __destruct() {
    //Utasitasok
}
```

- Példa:

```
class Animal {
    public $name = "No-name animal";

    public function __construct($name) {
        echo "I'm alive!<br>";
        $this->name = $name;
    }

    public function __destruct() {
        echo "I'm dead now :(";
    }
}

$animal = new Animal("Bob");
echo "Name of the animal: " . $animal->name. "<br>";
```

```
I'm alive!
Name of the animal: Bob
I'm dead now :(
```

Statikus tagok

- Alap esetben az osztály tagjai az objektumokhoz kapcsolódnak:
  - Annyi példányban léteznek, ahány objektum van az adott osztályból, és a tagváltozók értéke objektumonként eltérhet
- A statikus tagok az osztályhoz kapcsolódnak
  - Csak egy példány létezik belőlük az egész programban, erre hivatkozunk mindenhol
- Nem szerepelhet a *$this* kifejezés, a statikus metódusban

Statikus tagok - Példa


```php
class User {

    public $name;
    public $age;
    public static $minimumPasswordLength = 6;

    public function Describe() {
        return $this->name . " is " . $this->age . " years old";
    }

    public static function ValidatePassword($password) {
        if (strlen($password) >= self::$minimumPasswordLength)
            return true;
        else
            return false;
    }

}
```

```php
$password = "test";
if (User::ValidatePassword($password))
    echo "Password is valid!";
else
    echo "Password is NOT valid!";
```

Statikus tagok elérése

- Statikus tagok elérésére a hatókör (::) operátort használjuk
- A hatókör operátor bal oldalán szerepelhet egy osztálynév, illetve osztályon belüli használatkor a *self* és *parent* kulcsszó.
  - self: az osztályon belül önmagára vonatkozik
  - parent: az ősosztályra vonatkozik
- Példa:
  - ```
    if (strlen($password) >=
        self::$minimumPasswordLength) …
    ```
  - ```
    if (User::ValidatePassword($password)) …
    ```

15

Öröklődés

Öröklődés

- Az öröklődés lehetővé teszi, hogy egy osztály örökölje egy másik osztály *(szülőosztály)* tagjait, majd ezeket kiegészítse a saját tagjaival
- Megjegyzések:
  - Egy osztály csak egy szülőosztálytól örökölhet
  - Ha a szülő szintén örököl tagokat a saját szülőosztályától, akkor azokat is továbbadja
  - Ha az osztály egy tagjának neve megegyezik egy örökölt tag nevével, akkor az felülírja az örökölt tagot

Öröklődés

- Szintaxis:

```
class származtatott extends szülő {
   osztály_kódja
}
```

- Példa:

```
Class Shape{
        …
}
Class Circle extends Shape {
        …
}
```

Öröklődés – 1.Példa


```php
class Teglalap {
  public $magassag;
  public $szelesseg;

  public function __construct($szelesseg, $magassag) {
    $this->szelesseg = $szelesseg;
    $this->magassag = $magassag;
   }

   public function teruletSzamitas() {
    return $this->magassag * $this->szelesseg;
   }
 }
```

```php
class Negyzet extends Teglalap {
  public function __construct($meret) {
    $this->magassag = $meret;
    $this->szelesseg = $meret;
  }

  public function teruletSzamitas() {
    return pow($this->magassag, 2);
  }
}
```

```php
$obj = new Negyzet(7);
$a = $obj->teruletSzamitas();
echo $a;
```

Öröklődés – 2.Példa


```
class Shape {
  private $_color = "black";
  private $_filled = false;

  public function getColor() {
    return $this->_color;
  }

  public function setColor( $color ) {
    $this->_color = $color;
  }

  public function isFilled() {
    return $this->_filled;
  }

  public function fill() {
    $this->_filled = true;
  }
```

Öröklődés – 2.Példa


```php
class Circle extends Shape {
  private $_radius = 0;

  public function getRadius() {
    return $this->_radius;
  }

  public function setRadius( $radius ) {
    $this->_radius = $radius;
  }

  public function getArea() {
    return M_PI * pow( $this->_radius, 2 );
  }
}
```

```php
$myCircle = new Circle;
$myCircle->setColor( "red" );
$myCircle->fill();
$myCircle->setRadius( 4 );
echo "<h2>My Circle</h2>";
echo "<p>My circle has a radius of " . $myCircle->getRadius() . ".</p>";
echo "<p>The area of my circle is: " . $myCircle->getArea() . ".</p>";
```

Öröklődés – 2.Példa


```
class Tablazat {

    var $tablazatSorok = array();
    var $oszlopNevek = array();
    var $oszlopszam;

    function Tablazat($oszlopNevek) {
        $this->oszlopNevek = $oszlopNevek;
        $this->oszlopszam = count($oszlopNevek);
    }

    function ujSor($sor) {
        if (count($sor) != $this->oszlopszam)
            return false;
        array_push($this->tablazatSorok, $sor);
        return true;
    }
```

```
function ujNevesSor($asszoc_sor) {
    if (count($asszoc_sor) != $this->oszlopszam)
        return false;
    $sor = array();
    foreach ($this->oszlopNevek as $oszlopNev) {
        if (!isset($asszoc_sor[$oszlopNev]))
            $asszoc_sor[$oszlopNev] = "";
        $sor[] = $asszoc_sor[$oszlopNev];
    }
    array_push($this->tablazatSorok, $sor);
}

function kiir() {
    print "<pre>";
    foreach ($this->oszlopNevek as $oszlopNev)
        print "<B>$oszlopNev</B> ";
    print "\n";
    foreach ($this->tablazatSorok as $y) {
        foreach ($y as $xcella)
            print "$xcella ";
        print "\n";
    }
    print "</pre>";
}
```

Öröklődés – 2.Példa


```php
$proba = new Tablazat( array("a","b","c"));
$proba->ujSor( array(1,2,3));
$proba->ujSor( array(4,5,6));
$proba->ujNevesSor( array ( "b"=>0, "a"=>6, "c"=>3 ));
$proba->kiir();
```

a	b	c
1	2	3
4	5	6
6	0	3

Öröklődés – 2.Példa


```
class HTMLTablazat extends Tablazat{

    var $hatterSzin;
    var $cellaMargo = 2;

    function HTMLTablazat($oszlopNevek, $hatter = "#ffffff") {
        Tablazat::Tablazat($oszlopNevek);
        $this->hatterSzin = $hatter;
    }

    function cellaMargoAllit($margo) {
        $this->cellaMargo = $margo;
    }
```

Öröklődés – 2.Példa


```
function kiir() {
    print "<table cellPadding=\"$this->cellaMargo\"border=1>\n";
    print "<tr>\n";
    foreach ($this->oszlopNevek as $oszlopNev)
        print "<th bgColor=\"$this->hatterSzin\">$oszlopNev</th>";
    print "</tr>\n";
    foreach ($this->tablazatSorok as $sor => $cellak) {
        print "<tr>\n";
        foreach ($cellak as $cella)
            print "<td bgColor=\"$this->hatterSzin\">$cella</td>\n";
        print "</tr>\n";
    }
    print "</table>";
}
```

Öröklődés – 2.Példa


```
$proba = new HTMLTablazat( array("a","b","c"), "#00FF00");
$proba->cellaMargoAllit( 7 );
$proba->ujSor( array(1,2,3));
$proba->ujSor( array(4,5,6));
$proba->ujNevesSor( array ( "b"=>0, "a"=>6, "c"=>3 ));
$proba->kiir();
```

a	b	c
1	2	3
4	5	6
6	0	3

Absztrakt metódus, osztály

- Absztrakt metódus: üres metódus, mely csak örökítési célt szolgál
- Absztrakt osztály: absztrakt metódust tartalmazó, nem példányosítható osztály
- Szintaxis:

```
abstract class AbstractClass{
        abstract public function AbstractFunc();
        …
}
```

Absztrakt osztály – Példa


```php
abstract class Shape {
  private $_color = "black";
  private $_filled = false;

  public function getColor() {
    return $this->_color;
  }

  public function setColor( $color )
    $this->_color = $color;
  }

  public function isFilled() {
    return $this->_filled;
  }

  public function fill() {
    $this->_filled = true;
  }

  public function makeHollow() {
    $this->_filled = false;
  }

  abstract public function getArea();
}
```

```php
class Circle extends Shape {
  private $_radius = 0;

  public function getRadius() {
    return $this->_radius;
  }

  public function setRadius( $radius ) {
    $this->_radius = $radius;
  }

  public function getArea() {
    return M_PI * pow( $this->_radius, 2 );
  }
}
```

```php
$myCircle = new Circle;
$myCircle->setColor( "red" );
$myCircle->fill();
$myCircle->setRadius( 4 );
```

Polimorfizmus

- Polimorfizmus (polymorphism, többbalakúság)
  - Azt jelenti, hogy ugyanarra az üzenetre különböző típusú objektumok különbözőképpen reagálnak – minden objektum a saját metódusával

Polimorfizmus - Példa


```php
class Fruit {
  public function peel() {
    echo "<p>I'm peeling the fruit...</p>";
  }

  public function slice() {
    echo "<p>I'm slicing the fruit...</p>";
  }

  public function eat() {
    echo "<p>I'm eating the fruit. Yummy!</p>";
  }

  public function consume() {
    $this->peel();
    $this->slice();
    $this->eat();
  }
}
```

```php
class Grape extends Fruit {
  public function peel() {
    echo "<p>No need to peel a grape!</p>";
  }

  public function slice() {
    echo "<p>No need to slice a grape!</p>";
  }
}
```

Polimorfizmus - Példa


```
echo "<h2>Consuming an apple...</h2>";
$apple = new Fruit;
$apple->consume();

echo "<h2>Consuming a grape...</h2>";
$grape = new Grape;
$grape->consume();
```

Overriding Methods

Consuming an apple...

I'm peeling the fruit...

I'm slicing the fruit...

I'm eating the fruit. Yummy!

Consuming a grape...

No need to peel a grape!

No need to slice a grape!

I'm eating the fruit. Yummy!

Objektum osztályának vizsgálata

- Szintaxis:

  objektum instanceof *osztály*

- Visszaadja, hogy az objektum példánya-e a megadott osztálynak, vagy leszármazottjának
- Példa:

```php
class Animal {

    var $name;

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

}
```

```php
class Dog extends Animal {

    function speak() {
        return "Woof, woof!";
    }

    function playFetch() {
        return 'getting the stick';
    }

}
```

Objektum osztályának vizsgálata

- Példa (folytatás):

```php
class Cat extends Animal {

    function speak() {
        return "Meow...";
    }
}

$animals = array(new Dog('Skip'), new Cat('Snowball'));

foreach ($animals as $animal) {
    print $animal->name . " says: " . $animal->speak() . '<br>';
    if ($animal instanceof Dog)
        echo $animal->playFetch();
}
```

```
Skip says: Woof, woof!
getting the stickSnowball says: Meow...
```

Osztály és objektum függvények

- *get_class_vars()*, *get_object_vars()*: visszaadja egy osztály illtve egy objektum tulajdonságainak tömbjét
- *get_class()*: egy objektum osztályának a nevét adja meg
- *class_exists()*: megvizsgálja, hogy definiált-e az osztály
- *get_parent_class()*: visszaadja egy objektum vagy osztály szülő osztályát
- *is_a()*: megvizsgálja, hogy az objektum leszármazottja vagy tagja-e egy osztálynak
- *is_subclass_of()*: megvizsgálja, hogy egy objektum egy megadott osztálynak egy alosztályához tartozik-e

Metódus túlterhelés (overloading)

- Nem támogatja a PHP
- Megoldás: _call függvény használata
- Példa:

```php
class Calculate {

    public function __call($name, $arguments) {

        if ($name = 'sum') {

            if (count($arguments) === 2) {
                return $this->sum1($arguments[0], $arguments[1]);
            }
            if (count($arguments) === 3) {
                return $this->sum2($arguments[0], $arguments[1], $arguments[2]);
            }
        }
    }
```

Metódus túlterhelés (overloading)


```php
public function sum1($a, $b) {
    echo $a + $b;
}

public function sum2($a, $b, $c) {
    echo $a + $b + $c;
}
```

```php
$obj = new Calculate();
echo $obj->sum(1, 2);
echo "<br>";
echo $obj->sum(1, 2, 3);
```

Projekt témák

WordPress projekttéma javaslatok

- **Webshop bővítmányek:**
  - WP e-Commerce Plugin
  - WP Online Store
  - Woocommerce Plugin
  - eShop Plugin
  - Cart66 Lite
  - Jigoshop WordPress e-commerce Plugin
  - Quick Shop Plugin
  - YAK shopping cart Plugin
  - Zingiri WebShop v 2.5.9 magyar fordítás
- Leírás magyarul a fenti bővítményekről:

  http://efrud.hu/a-nagy-wordpress-webaruhaz-bovitmeny-szemle/

WordPress projekttéma javaslatok

- **Étterem sablon:**
  - Confit (http://theme.wordpress.com/themes/confit/)
  - Easy Restaurant Menu Manager
  - WPPizza
  - ReDi Restaurant Reservation

  A témával kapcsolatosan itt lehet még olvasni:

  http://www.moeseo.com/10-best-wordpress-plugins-for-a-restaurant-website/
- **Online rendelés/kölcsönzés:**
  - Checkfront Online Booking System
  - Rezgo Online Booking

  A témával kapcsolatosan itt lehet még olvasni:

  http://www.wpmayor.com/plugin-reviews/best-wordpress-booking-plugins/

Más témák

- **Közgáz alapú téma webes megvalósítása (lízing, mérleg készítés, hitel kalkulátor, stb)**
- **Online-bank szimulátor**
- **Befektetés kezelő rendszer megvalósítása**
- **Házi feladat beküldő/ellenőrző rendszer**
- On-line jelenléti napló
- Diákok nyilvántartása bentlakásban
- Raktárkezelő webes alkalmazás elkészítése
- PHP alapú webáruház megvalósítása egy gyakorlati példán keresztül
- Használtautó-kereskedés nyilvántartó rendszere
- Online munkaközvetítő készítése
- Gépjármű nyilvántartó PHP-ben