Kurs WWW

Paweł Rajba

pawel@ii.uni.wroc.pl
http://pawel.ii.uni.wroc.pl/

Spis treści

Wprowadzenie

- Deklaracja klasy
  - ```
    class Nazwa {
      // pola
      // metody
    }
    ```
- Nazewnictwo plików i składowanie
  - Każda klasa powinna być w osobnym pliku
  - Nazwa pliku
    - Taka sama jak nazwa klasy (**`{NazwaKlasy}.php`**)
    - Z prefiksem class (**`class.{NazwaKlasy}.php`**)


    ```

# Wprowadzenie

- Nazewnictwo klas
  - Nazwa powinny być z wielkiej litery
  - Można poprzedzić odpowiednim prefiksem
    - C – klasy
    - CS – klasy statyczne
    - I – interfejsy
  - W nazwie klasy nie mają znaczenia małe i wielkie litery (np. nie można zadeklarować klas a i A)
    - Ale funkcja get_class($klasa) zwróci wartość poprawną np. dla Bob zwróci Bob (inaczej niż w PHP4, tam zwróciłoby bob)

 

# Wprowadzenie

- Obiekty tworzymy za pomocą operatora **new**

```
<?php
  require_once("class.CDemo.php");
  $demo=new CDemo();
?>
```

Automatyczne ładowanie

- Dawniej, na początku każdego skryptu był stos instrukcji require do załadowania plików z klasami
- W PHP5 jest mechanizm automatycznego ładowania pliku z klasą, gdy klasa jest użyta
  - Mechanizm polega na wykonaniu funkcji __autoload, która ma jeden parametr – nazwa klasy
  - Pytanie co wtedy, gdy chcemy hierarchizować strukturę katalogów, w której składowane są klasy
    - Kwestia nazewnictwa
    - Wyszukiwanie danego pliku (może być nieefektywne)
- Przykład: autoload.php

Składowe klasy

- Zmienne klasy
  - Dostęp do zmiennych: $this->zmienna;
  - Zaleca się tworzenie metod dostępowych do zmiennych klasy (enkapsulacja danych):
    - set{NazwaZmiennej} i get{NazwaZmiennej}
- Stałe klasy
  - Deklaracja: **`const nazwa_stalej`**
  - Dostęp: **`self::nazwa_stałej`**
  - Przykład:

```
const PI = 3.14;
echo self::constant . "\n<br>";
```


Składowe klasy

- Metody
  - Deklaracja metody jest podobna deklaracji funkcji

```
class Demo {
  function drukowanie( $zmienna )  {  }
}
```

  - Parametry metod mogą pobierać typy argumentów (tylko dla typów obiektowych)

```
class Demo {
  function drukowanie( Napis $zmienna ) { }
}
```


Składowe klasy

- Składowe statyczne
  - Tworzymy przez dodanie słowa static
  - Związane z klasą nie z obiektami
  - Dostęp jest poprzez operator ::
  - Przykłady odwołań
    - self::$zmienna
    - self::metod()
    - COsoba::$imie
    - COsoba::getLiczbaOsob()

Składowe klasy

- Składowe statyczne, przykład
  - ```
    class CDatabase {
      private static $db=NULL;
      __construct( $host, $user, $pass ) {
        if ( self::$db == NULL ) {
          self::$db=dbConnect( $host, $user, $pass );
        }
      }
    ```
- Składowe finalne
  - Deklarujemy przez dodanie słowa final
  - Finalne mogą być klasy i metody
    - Po takich klasach nie można dziedziczyć
    - Metod nie można nadpisywać w klasach potomnych


    ```

# Widoczność składowych

- Mamy do dyspozycji trzy specyfikatory dostępu:
  - public – dostępne zewsząd
  - protected – dostępne tylko w klasach potomnych
  - private – dostępne tylko dla klasy w której występuje

 

# Konstruktory i tworzenie obiektów

- Deklaracja konstruktora
  - __construct( $parametry )
- Nie ma przeciążania konstruktorów
- Można używać również konwencji nazw z PHP4
  - wyższy priorytet ma nowa konwencja (__construct)
- Wywoływanie
  - Jeśli jest konstruktor, nie są wywoływane konstruktory klas bazowych
  - Jeśli nie ma konstruktora, są wywoływane konstruktory klas bazowych

 

# Destruktory i likwidacja obiektów

- Deklaracja destruktora
  - __destruct()
- Obiekt jest usuwany gdy
  - zakończy się skrypt
  - zmienna znika z bieżącego zakresu
  - zmiennej jawnie przypiszemy wartość pustą
  - usuniemy zmienną za pomocą funkcji unset()
    - usuwanie jest na podstawie tzw. licznika odwołań do obiektu
- Funkcja destruct jest bezparametrowa

 

# Destruktory i likwidacja obiektów

- Wywoływanie
  - Jeśli jest destruktor, nie są wywoływane destruktory klas bazowych
  - Jeśli nie ma destruktora, są wywoływane destruktory klas bazowych
- Destruktory przydają głównie do
  - zamknięcia połączeń do baz danych
  - zamknięcia otwartych plików
  - innych prac porządkowych
- Przykłady: zycie-obiektow.php, destruktory.php

 

# Przypisania i klonowanie

- Wszystkie przypisania obiektów w PHP5 są przez referencje (inaczej, niż w PHP4)
- Żeby sklonować obiekt, należy użyć instrukcję clone:
  - $nowy=clone $stary
- Przykład: klonowanie.php

 

# Dziedziczenie

- Składnia
  - class Bazowa { }
    class Potomna extends Bazowa { }
- Komentarz
  - Dziedziczyć można z tylko jednej klasy
  - Dostępny jest mechanizm polimorfizmu
  - Operator instanceof pozwala ustalić, czy obiekt jest danej klasy
  - Dostęp do nadpisanych składowych jest poprzez operator parent::

 

# Dziedziczenie

- Przykłady
  - dziedziczenie1.php
  - dziedziczenie2.php
  - polimorfizm.php

 

# Klasy abstrakcyjne

- Klasę taką deklarujemy dodając słowo abstract
  - abstract class K { ... }
- Klasa abstrakcyjna powinna mieć co najmniej jedną metodą abstrakcyjną
  - protected abstract function metoda()
- Metody nie mogą być private lub final
- Nie można tworzyć obiektów klas abstrakcyjnych

 

# Klasy abstrakcyjne

- Przykład
  - ```
    abstract class CDatabase
    {
      abstract public function connect();
      abstract public function query();
      abstract public function fetch();
      abstract public function close();
    }
    ```
  - ```
    class CMySQL extends CDatabase
    {
      ...
    }
    ```


Interfejsy

- Interfejs zbiór deklaracji metod publicznych
- Deklaracja

```
interface INazwa {
   public function f1($arg); ...
}
```

  - Nazwa
- Później klasa może implementować interfejs

```
class K implements INazwa { ... }
```

  (musi defininiować wszystkie metody interfejsu)
- Klasa można implementować kilka interfejsów

Interfejsy

- Nazewnictwo plików
  - I{nazwainterfejsu}.php
  - interface.{nazwa_interfejsu}.php
- Przykład: interfejsy.php

Metody magiczne

- Metody __get() i __set()
  - Zostaną wykonane w przypadku pobrania/przypisania wartości nieistniejącej zmiennej klasy
  - Można wykorzystać do utworzenia funkcjonalności setterów i getterów
    - Definiowanie metod setXXX i getXXX dla 20 pól może być męczące
    - Przedstawione w przykładzie rozwiązanie niestety ma poważne wady
- Przykład: magiczne1.php

Metody magiczne

- Metoda __call()
  - Przechwytuje wywołania metod, które w klasie nie zostały zdefiniowane
  - Przydatne do implementacji mechanizmu agregacji
- Metoda __clone()
  - Jest wykonywana podczas klonowania obiektu
- Przykład: magiczne2.php

Metody magiczne

- Metoda __toString
  - Zwraca napis kiedy nazwa klasy jest traktowana jako napis (np. argument echo lub print)
  - Nie działa w następujących przypadkach
    - Przy łączeniu napisów operatorem .
    - Gdy obiekt jest umieszczany w cudzysłowie lub składni heredoc
    - Gdy obiekt jest traktowany w funkcji printf jako napis (%s)
    - Gdy obiekt przekazywany do funkcji jako parametr (w przypadku, gdy funkcja wymaga parametru typu string)
- Przykład: magiczne3.php

Porównywanie obiektów

- Obiekty możemy porównywać operatorami
  - == – porównywane obiekty będą równe, gdy
    - są instancjami tej samej klasy
    - wartości wszystkich pól są takie same
  - === – porównywane obiekty będą równe, gdy
    - zmienne wskazują na tą samą instację tej samej klasy
- Przykład: pogmatwany.php

