Obiektowy PHP

Czym jest obiekt?

W programowaniu obiektem można nazwać każdy abstrakcyjny byt, który programista utworzy w pamięci komputera. Jeszcze bardziej upraszczając to zagadnienie, można powiedzieć, że obiekt to coś, co może przechowywać dane oraz wykonywać różne operacje. Trzeba także wiedzieć, że jeden obiekt może zawierać inne obiekty – dokładnie tak jak w świecie realnym. Przecież wewnątrz przykładowego samochodu mamy np. fotele, kierownicę, silnik itp.. Nie można jednak tworzyć obiektów bez ich wcześniejszego zaplanowania. Czy da się bowiem skonstruować dom bez planu? Raczej nie. Tak jest też w PHP – aby utworzyć obiekt, najpierw trzeba go zaprojektować, utworzyć jego plan. Taki plan obiektu nazywamy klasą. Określa ona, jak będą zbudowane obiekty tworzone na jej podstawie. Innymi słowy, klasa to zdefiniowany przez programistę nowy typ danych.

Definicja klasy

Klasa jest definiowana za pomocą słowa kluczowego class. Ogólna postać takiej konstrukcji jest następująca:

```
class nazwa_klasy
{
//definicja klasy
}
```

Klasa może zawierać nieograniczoną liczbę zmiennych (nazywanych polami, lub właściwościami) oraz funkcji (nazywanych metodami) – te elementy nazywamy składowymi klasy.

Składowe klasy – pola i metody

Obiekty mogą przechowywać dane oraz wykonywać zadania. Dane mogą być przechowywane w zmiennych, które umieszczamy w projekcie obiektu, czyli w klasie. Takie zmienne nazywamy polami klasy. Aby obiekt mógł wykonywać zadania, potrzebny jest z kolei kod wykonywalny. Ten kod umieszczamy w funkcjach przynależnych do klasy. Takie funkcje nazywamy metodami klasy.

Definicja pola klasy ma następującą postać:

```
class nazwa_klasy
{
public $nazwa_pola;
}
```

Metody klasy tworzy się podobnie jak zwykłe funkcje, używając słowa kluczowego function. Typowa definicja będzie miała postać:

class nazwa_klasy

{

Public function nazwa_metody(argumenty)

{

//wnętrze metody

}

}

Aby lepiej przećwiczyć tego typu konstrukcje, utworzymy teraz prostą klasę o nazwie Uzytkownik, której zadaniem będzie obsługa danych związanych z użytkownikami serwisu internetowego. Będzie ona zawierała tylko dwa pola:

- nazwa – przechowuje nazwę użytkownika
- id – przechowuje identyfikator użytkownika.

```php
<?php
class Uzytkownik
{
  public $nazwa, $id;
}
?>
```

Do klasy Uzytkownik dodajmy teraz cztery metody:

- ustawNazwe – zmienia wartość pola $nazwa;
- pobierzNazwe – pobiera wartość pola $nazwa;
- ustawId – zmienia wartość pola $id;
- pobierzId – ustawia wartość pola $id.

```php
<?php
class Uzytkownik
{
  public $nazwa, $id;
  public function ustawNazwe($wartosc)
  {
    $this->nazwa = $wartosc;
  }
  public function pobierzNazwe()
  {
    return $this->nazwa;
  }
  public function ustawId($wartosc)
  {
    $this->id = $wartosc;
  }
  public function pobierzId()
  {
    return $this->id;
  }
}
?>
```

Tworzenie obiektów

Gdy mamy zdefiniowaną klasę (czyli nowy typ danych), możemy na jej podstawie tworzyć obiekty. Mówi się, że obiekt jest instancją klasy. Aby go utworzyć, należy użyć słowa ***new***, po którym musi wystąpić nazwa klasy zakończona nawiasem :

new nazwa_klasy():

Aby jednak można było skorzystać z tak utworzonego obiektu, należy go przypisać do jakiejś zmiennej:

$nazwa_zmiennej= new nazwa_klasy();

Po takim przypisaniu ***$nazwa_zmiennej*** pozwala na odwoływanie się do nowego obiektu typu ***nazwa_klasy***.

Odwołania do składowych

Aby odwołać się do dowolnej składowej klasy, należy użyć operatora ->. A zatem dla pól takie odwołanie będzie miało postać:

$nazwa_obiektu->nazwa_pola;

A dla metod:

$nazwa_obiektu->nazwa_metody(argumenty_metody);

W przypadku pól w ten sposób można zarówno odczytywać, jak i zapisywać ich wartości. Jeśli na przykład zmienna o nazwie ***$user*** zawiera obiekt klasy ***Uzytkownik***, to aby odczytać wartość pola nazwa i przypisać ją zmiennej $wartość, należy wykonać instrukcję:

$wartość= $user->nazwa;

Aby natomiast polu id tego obiektu przypisać wartość 123 – instrukcję:

$user->id=123;

Sposób odwołania się do składowych klasy ***Uzytkownik***:

```php
<?php
class Uzytkownik
{
  public $nazwa, $id;
  public function ustawNazwe($wartosc)
  {
    $this->nazwa = $wartosc;
  }
  public function pobierzNazwe()
  {
    return $this->nazwa;
  }
  public function ustawId($wartosc)
  {
    $this->id = $wartosc;
  }
  public function pobierzId()
  {
    return $this->id;
  }
}

$user1 = new Uzytkownik();
$user2 = new Uzytkownik();

$user1->nazwa = 'j.kowalski';
$user2->ustawNazwe('a.nowak');

echo $user1->nazwa;
echo "<br />";
echo $user2->pobierzNazwe();
?>
```

Tworzenie konstruktorów

Po utworzeniu obiektu jego pola są puste, nie zawierają żadnych wartości. Często nie jest to sytuacja pożądana i należy nadać im pewne wartości początkowe. Załóżmy na przykład, że w przypadku znanej nam już klasy ***Uzytkownik*** chcielibyśmy, aby utworzony obiekt tego typu był wypełniony wartościami domyślnymi. Pole nazwa powinno przyjąć wartość Gość, a pole id wartość 0. Do tego celu obiektowe języki programowania udostępniają mechanizm konstruktorów.

Konstruktory są to specjalne metody wykonywane podczas tworzenia obiektów danej klasy. Aby zdefiniować konstruktor, należy umieścić w klasie metodę o nazwie ***__construct***, schematycznie:

```
class nazwa_klasy
{
function__construct(
)
{
//treść konstruktora
}
//pozostałe składowe
        }
```

Jeśli zatem chcemy, aby obiekty klasy ***Uzytkownik*** przyjmowały automatycznie domyślne wartości pól, do klasy należałoby dodać konstruktor w postaci:

```php
<?php
class Uzytkownik
{
  public $nazwa, $id;
  function __construct()
  {
    $this->nazwa = "Gość";
    $this->id = 0;
  }
  public function ustawNazwe($wartosc)
  {
    $this->nazwa = $wartosc;
  }
  public function pobierzNazwe()
  {
    return $this->nazwa;
  }
  public function ustawId($wartosc)
  {
    $this->id = $wartosc;
  }
  public function pobierzId()
  {
    return $this->id;
  }
}
?>
```

Aby się przekonać, że konstruktor faktycznie działa wystarczy wykonać kod skryptu:

```php
<?php
include "Uzytkownik.php";

$user1 = new Uzytkownik();

echo $user1->nazwa;
echo "<br />";
echo $user1->id;
?>

```

Parametry konstruktorów

Konstruktory nie muszą być bezargumentowe, mogą również przyjmować argumenty, które zostaną użyte na przykład do zainicjowania pól obiektu.

class nazwa_klasy

{

function_construct(argument1, argument2, … argumentN)

{

//treść konstruktora

}

}

Jeśli konsruktor przyjmuje argumenty, przy tworzeniu obiektu należy je oczywiście podać, czyli zamiast stosownej do tej pory konstrukcji:

$zmienna= new nazwa_klasy()

trzeba zastosować wywołanie:

$zmienna= new nazwa_klasy(argument1, argument2, … argumentN)

W przypadku używanej do tej pory przykładowej klasy ***Uzytkownik*** przydatny byłby np. konstruktor, który przyjmowałby dwa argumenty oznaczające nazwę i identyfikator użytkownika.

```
function __construct($nazwa, $id)
{
  $this->nazwa = $nazwa;
  $this->id = $id;
}
```

Wykorzystanie konstruktora dwuargumentowego:

```
1 <?php
2 include "Uzytkownik.php";
3
4 $user1 = new Uzytkownik("j.kowalski", 15);
5
6 echo $user1->nazwa;
7 echo "<br />";
8 echo $user1->id;
9 ?>
```

Destruktory

Destruktory to przeciwieństwo konstruktorów. Są to metody wykonywane wtedy, gdy obiekt jest niszczony, usuwany z pamięci. Są przydatne, jeśli na przykład klasa używa zasobów systemowych, które muszą być zwolnione przed usunięciem obiektu. Otóż, podobnie jak w przypadku konstruktorów, jest to specjalna funkcja o nazwie ***__destruct***, którą należy umieścić w kodzie klasy.

```
class nazwa_klasy
{
function __destruct(
)
    {
    //treść konstruktora
    }
//pozostałe składowe
```

Automatyczne ładowanie kodu klasy

Definicje klas często przechowuje się w osobnych plikach o nazwach zgodnych z nazwą klasy, np. klasa ***Uzytkownik*** była przechowywana w pliku o nazwie ***Uzytkownik.php***. Aby użyć takiej definicji w skrypcie, konieczne jest dołączenie treści pliku za pomocą instrukcji ***include***. Jeśli jednak klas jest dużo, niezbędne staje się używanie wielu instrukcji ***include***. Ten problem można łatwo rozwiązać, stosując technikę automatycznego ładowanie pliku z kodem klasy w momencie pierwszego odwołania do klasy. Używa się do tego celu funkcji ***__autoload***.

```php
<?php
function __autoload($nazwa_klasy){
  include $nazwa_klasy . '.php';
}
$user1 = new Uzytkownik("j.kowalski", 15);

echo $user1->nazwa;
echo "<br />";
echo $user1->id;
?>
```

Obiektowa lista odwiedzin

Wiedza o tym kto odwiedza naszą witrynę, niewątpliwie bardzo pomaga w jej prowadzeniu. Z pewnością będzie więc przydatny skrypt, który udostępnia tego typu informacje. Takie zadanie

zrealizujemy w poniższym skrypcie. W bazie danychzapisywane są dane dotyczące:

- daty i czasu
- odwiedzin, adresu IP,
- typu przeglądarki.

Klasa ***Stats*** obsługiwać będzie połączenie z bazą MySQL i będzie zawierać następujące elementy:

1. Prywatne pole przechowujące identyfikator połączenia z serwerem MySQL.

2.Konstruktor do którego w formie parametrów przekazujemy:

- adres serwera MySQL

-użytkownika

-hasło

-nazwę bazy

oraz zwracał będzie powodzenie połączenia z bazą.

3. Metodę wstawiania rekordów do tabeli.

Metoda jako parametr pobiera zmienną string (zapytanie SQL typu insert) natomiast zwraca liczbę wierszy w bazie, na które zapytanie miało wpływ lub false w razie niepowodzenia.

4. Metodę pobierania rekordów z bazy.

Metoda jako parametr pobiera zmienną string (zapytanie SQL typu select) natomiast zwraca tabelę z rekordami lub false w razie niepowodzenia.

5. Destruktor klasy kończący połączenie z bazą