PHP

Programowanie PHP

Remigiusz Pyrek

Zmienne (*vars*)

Konstrukcja programistyczna posiadająca minimum dwa podstawowe atrybuty: symboliczną nazwę oraz wartość. Za pomocą nazwy możemy w kodzie źródłowym odwoływać się do jej wartości. Nazwa zmiennej może być dowolna i służy do identyfikowania zmiennej dlatego często nazywana jest identyfikatorem.

Zmienne w języku PHP

Każdą zmienną w PHP zapisuje się, poprzedzając jej nazwę znakiem dolara "`$`". Wielkość liter w nazwie zmiennej jest rozróżniana. Poprawne nazwy zmiennych zaczynają się od litery lub znaku podkreślenia. Długość nazwy zmiennej jest dowolna.

```
$imie = "Jan";
$IMIE = "Piotr";
$4you = "4ever";          // błędna nazwa zmiannej
$_4you = "4ever";         // poprawna nazwa zmiennej
```

Metody tworzenia długich nazw zmiennych

Długie nazwy zmiennych tworzymy poprzez rozdzielenie pojedynczych fraz znakiem podkreślenia.

```
$imie_i_nazwisko = "Jan Kowalski";
$uzytkownik_zalogowany = true;
```

Typowanie dynamiczne

Typowanie zmiennej w języku PHP odbywa się na podstawie jej wartości. Wartości zmiennych typu string zapisujemy zawsze wewnątrz znaków apostrof `'wartość tekstowa'` lub cudzysłów `"wartość tekstowa"`

```
$a = "Jan Kowalski";      // zmienna typu string
$b = 1;                   // zmienna typu int
$c = 36.6;                // zmienna typu float
$d = true;                // zmienna typu boolean
```

Uwaga *Wielkość liter w nazwach zmiennych jest rozróżniana, dlatego zmienna `$IMIE` oraz `$imie` są sobie różne. Nazwa `$this` jest zmienną spacjalną i nie można przypisywać jej wartości. Nazwy zmiennych nie mogą rozpoczynać się od cyfry. W nazwach zmiennych można stosować znaki diaktryczne.*

Operatory (*operators*)

Operator jest czymś, co przyjmuje jedną lub więcej wartości i daje inną wartość w taki sposób, że konstrukcja sama w sobie staje się wyrażeniem. Operatory są pogrupowane w zależności od ilości podjętych przez nich wartości.

Operatory przypisania

Podstawowym operatorem przypisania jest "=" na przykład `$a = 10`. Oznacza to, że lewy operand dostaje wartość wyrażenia po prawej stronie (czyli „ustawia się na").

```
$a = 5+5*5;
$a += 2;

$b = "Jan ";
$b .= "Kowalski";
```

Operatory arytmetyczne

Operatory arytmetyczne działając na operandach zwracają wyliczoną wartość liczbową, realizując podstawowe operacje arytmetyki.

```
$a + $b;      // operator dodawania
$a - $b;      // operator odejmowania
$a * $b;      // operator mnożenia
$a / $b;      // operator dzielenia
$a % $b;      // operator modulo
```

Operatory porównania

Operatory porównania, jak sugerują ich nazwy, pozwalają porównywać dwie wartości.

```
$a == $b;     // czy wartości są równe
$a === $b;    // czy wartości są identyczne
$a != $b;     // czy wartości nie są równe
$a <> $b;     // czy wartości nie są równe
$a !== $b;    // czy wartości nie są identyczne
$a < $b;      // czy wartości $a jest mniejsza od $b
$a > $b;      // czy wartości $a jest większa od $b
$a <= $b;     // czy wartości $a jest mniejsza lub równa $b
$a >= $b;     // czy wartości $a jest większa lub równa $b
```

Uwaga *Operator dzielenia zwraca zawsze wartość typu float nawet wtedy, kiedy operandy były typu integer. Operandy modulo konwertowane są na wartości typu integer przed przetwarzaniem.*

Operatory c.d. (*operators*)

Operator jest czymś, co przyjmuje jedną lub więcej wartości i daje inną wartość w taki sposób, że konstrukcja sama w sobie staje się wyrażeniem. Operatory są pogrupowane w zależności od ilości podjętych przez nich wartości.

Operatory tekstowe

Istnieją dwa operatory tekstowe. Pierwszym jest operator łączenia (" . "), który zwraca konkatenację jego prawych i lewych argumentów. Drugi to operator przypisania do łączenia (" .="), który dołącza argument po prawej stronie do argumentu po lewej stronie.

```
$a = "Jan";
$b = $a . "Kowalski";      // $b zawiera "Jan Kowalski"

$a = "Jan ";
$a .= "Kowalski";          // $a zawiera "Jan Kowalski"
```

Operatory zwiększenia i zmniejszenia

Istnieją dwa operatory inkrementacji i dekrementacji.

```
$a++;   // zwiększ wartość $a o 1 po wykonaniu działania
$a--;   // zmniejsz wartość $a o 1 po wykonaniu działania
++$a;   // zwiększ wartość $a o 1 przed wykonaniu działania
--$a;   // zmniejsz wartość $a o 1 przed wykonaniu działania
```

Operatory logiczne

Operatory logiczne działając na operandach zwracają wartość logiczną, realizując podstawowe operacje albegry Boole'a.

```
$a and $b;    // wartość $a i $b są prawdziwe
$a or $b;     // wartość $a lub $b są prawdziwe
$a xor $b;    // wartość $a lub $b są prawdziwe ale nie obie
!$a;          // wartości $a nie jest prawdziwa
$a && $b;     // wartość $a i $b są prawdziwe
$a || $b;     // wartość $a lub $b są prawdziwe
```

Operatory wykonania

PHP obsługuje jeden operator wykonawczy: backticks (``), nie są to pojedyncze cytaty! Interrpreter wykona zawartość backtików jako polecenie powłoki; wynik polecenia `ls` zostanie przypisany do zmiennej `$output`.

```
$output = `ls -al`;
echo "<pre>$output</pre>";
```

Wyrażenia (*expressions*)

Wyrażenia są najważniejszymi elementami składowymi PHP. Najprostszą i najdokładniejszą definicją wyrażenia jest „wszystko co ma wartość".

Wyrażenie proste

Najbardziej podstawową formą wyrażeń są stałe i zmienne. Jeśli napiszesz "`$a = 5;`" przypisujesz `5` do `$a`. Przypisane `5` ma wartość 5, lub innymi słowy `5` jest wyrażeniem o wartości 5. Podsumowując `$a` jest wyrażeniem o wartości `5`.

```
$a = 5;
$b = $a;
$b = 5;
```

Wyrażenia porównania

Popularnym typem wyrażeń w PHP są porównania. Wyrażenia te zwracają wartość `FALSE` lub `TRUE`. Wyrażenia te są powszechnie używane przy sprawdzaniu warunków, jak na przykład instrukcje `if`.

```
$a = 5; $b = 4;
if ($a>$b) {
    echo "$a większa od $b";
} else {
    echo "$a mniejsza od $b";
}
```

Wyrażenia inkrementacji i dekrementacji

Kolejnym przykładem korzystania z wyrażeń jest pre- i postinkrementacja, a także dekrementacja. Pre- oraz postinkrementacja zwiększają wartość zmiannej, różnica jest w wartości wyrażenia inkrementacji. Preinkrementacja `++$a` zwraca zwiększoną wartość, a postinkrementacja `$a++` zwraca orginalną wartość `$a`, a jej wartość zostanie zwiększona zaraz po jej odczytaniu, stąd nazwa post-inkrementacja.

```
$a = 5; echo $a++;
$b = 5; echo ++$b;
```

Wyrażenia złożone

Bardziej złożonymi wyrażeniami są funkcje. Można założyć, w poniższym przykładzie, że napisanie "`$c = foo();`" jest równoznaczne z napisaniem "`$c = 5`", ponieważ funkcje są wyrażeniami o wartości którą zwracają.

```
function foo(){
    return 5;
}
$c = foo();
```

Funkcje (*functions*)

Podprogram składowy, którego zadaniem jest działanie na rzecz określonych elementów danej klasy lub klas z nią spokrewnionych poprzez dziedziczenie.

Funkcje definiowane przez użytkownika

Nazwy funkcji obowiązują identyczne zasady, jak w przypadku wszystkich innych etykiet w PHP. Poprawna nazwa funkcji zaczyna się od litery lub podkreślnika, po których następuje dowolna ilość liter, cyfr i podkreślników.

```
function foo(){
   echo "Hello World!";
}
```

Argumenty funkcji

Dane mogą być przekazywane do funkcji przez listę argumentów, która jest listą oddzielonych przecinkami wyrażeń. Możliwe jest użycie różnej ilości argumentów dla funkcji definiowanych przez użytkownika. Wartości argumentów mogą być takę pobierane są za pomocą funkcji *func_num_args()*, *func_get_arg()* i *func_get_args()*.

```
function foo($text) {
   echo $text;
}
```

Domyślna wartość argumentu

Domyślna wartość musi być stałym wyrażeniem, a nie na przykład zmienną, członkiem klasy czy wywołaniem funkcji.

```
function foo($text = "Hello World!") {
   echo $text;
}
```

Zwracanie wartości

Wartości zwracane są przy użyciu opcjonalnego wyrażenia return. Wszystkie typy mogą być zwracane, łącznie z tablicami i obiektami. Powoduje to natychmiastowe zakończenie wykonywania funkcji i wznowienie wykonywania skryptu od linijki w której funkcja została wywołana.

```
function sum($a, $b) {
   return $a+$b;
}
echo sum(4,8);
```

Klasy i obiekty (*class*)

W programowaniu obiektowym klasa jest rozszerzalnym szablonem kodu programu do tworzenia obiektów, zapewniającym początkowe wartości dla stanu (zmiennych składowych) oraz implementacjami zachowań (funkcje lub metody członkowskie).

Podstawowa definicja klasy

Podstawowa definicja klasy zaczyna się od słowa kluczowego class, po którym następuje nazwa klasy, a następnie nawiasy klamrowe, które obejmują definicje, właściwości i metody należące do klasy.

```
class userStatus {
   public function displayWelcome(){
      echo "Hello World!";
   }
}
```

Właściwości klas

Zmienne klasy nazywane są „właściwościami" które są definiowane przy użyciu jednego ze słów kluczowych `public`, `protected` lub `private`, po którym następuje zwykła deklaracja zmiennej.

```
class userStatus {
   public $text = "Hello World!";
   private $status = true;
}
```

Konstruktory i destruktory

PHP 5 umożliwia programistom deklarowanie metod konstruktorskich dla klas. Klasy, które mają metodę konstruktora, wywołują tę metodę za każdym razem kiedy tworzony jest na nowo obiekt.

```
class userStatus {
   function __construct() {
      print "Wywołanie konstruktora klasy";
   }
}
$user = new UserStatus();
```

Metoda destruktora zostanie wywołana, gdy tylko nie ma innych odniesień do określonego obiektu lub podczas sekwencji zamykania.

```
class userStatus {
   function __destruct() {
      print "Wywołanie destruktora klasy";
   }
}
```