Cuprins

telefon: 0730 643 130
e-mail: office@atelierelеilbah.ro

Curs 02

Introducere în PHP. Tipuri de date

2.1 Comentarii în PHP

Un comentariu în PHP reprezintă o linie sau o secvență de cod care nu este executată ca parte a unui program. Scopul acestuia este de a oferi informații celui care se ocupă de dezvoltarea aplicației.

PHP suportă mai multe tipuri de comentarii, prezentate mai jos:

```
// Acesta este un comentariu de linie.

# Acesta este tot un comentariu de linie.

/*
Acesta este un comentariu
care ocupă
mai multe linii
*/
```

Pe lângă documentarea codului, comentariile pot fi folosite în procesul de dezvoltare pentru a testa anumite soluții prin omiterea unor părți din cod:

```
$x = 5 /* + 15 */ + 5;
echo $x; // 10
```

2.2 Ce este un tip de date?

În informatică și programare, un tip de date sau pur și simplu „tip”, este o clasificare ce identifică diferitele tipuri de date, cum ar fi: numere reale, numere întregi sau valori logice (adevărat sau fals); valori posibile pentru acest tip, operațiunile care pot fi efectuate pentru valorile de acest tip, sensul datelor și modul în care valorile de acest tip pot fi stocate.

2.3 Tipuri de date

În PHP există următoarele tipuri de date:

- Scalare: **int** (integer), **float**, **string**, **bool** (boolean);
- Compuse: **array**, **object**;
- Speciale: **resource**, **NULL**.

2.3.1 Tipuri de date scalare

Tipurile de date scalare pot avea o singura valoare. Din această categorie fac parte următoarele tipuri de date:

Int sau **integer** (număr întreg)
Este reprezentat de numere precum 1, 25, 42 sau 2006, respectiv valori negative precum -5, -100, -2022. Numerele întregi nu trebuie să conțină virgulă sau punctuație, sau orice fel de separatori.

```
$a = -4; // a este int
```

Float (numerele reale)
Tip de date folosit pentru a reprezenta numerele cu zecimale.

```
$a = -2.3; // a este float
```

Obs.: PHP nu permite folosirea virgulei pentru separarea zecimalelor.

String (sir)
Valori ce reprezintă șir de caractere. Se scriu între apostrof sau ghilimele.

```
$a = 'acasa';      // a este string
$b = "la scoala"; // b este string
```

Obs.: Poate fi scurt, fără caractere, cunoscut ca și șir gol.

```
$a = '';  // a este un șir gol
$b = "";  // b este un șir gol
```

Bool sau **boolean** (logic)
Acest tip de date poate avea doar două valori, **adevărat** sau **fals**. Aceste valori sunt cuvintele cheie **TRUE**, respectiv **FALSE** și sunt case-insensitive (pot fi scrise cu litere mici sau mari fără a afecta rezultatul).

```
$a = true; // a este bool
$b = TRUE; // b este bool
```

2.3.2 Tipuri de date compuse

Variabilele ce au acest tip sunt formate din informații compuse, fiecare informație putând avea alt tip de date. Din această categorie fac parte următoarele tipuri de date: **array** și **object**.

Array (tablou, șir)
Acest tip de date este compus din mai multe elemente. Fiecare element este o pereche formată dintr-o cheie și o valoare. Cheile pot fi valori întregi sau șiruri de caractere, în timp ce valorile pot avea orice tip de date.

```
$a= [1, 2, 3, "verde", "albastru", null, true];        // a este un tablou
$b = array(1, 2, 3, "verde", "albastru", null, true); // b este un tablou
```

Obs.: Exista posibilitatea sa avem un tablou gol.

```
$a = []; // a este tablou gol
```

Object (obiect)
Un obiect este un tip de date mai complex, capabil să stocheze și să manipuleze valori. Poate conține o grupare de mai multe tipuri de date (proprietăți), cât și metode (funcționalități).

```
class Masina
{
}
$m = new Masina(); // m este un obiect
```

2.3.3 Tipuri de date speciale

Resource (resursa)
Când PHP se conectează cu o sursă externă, precum un fișier sau o bază de date, stochează o referință către aceasta ca și resursă.

NULL (gol)
Tip de date ce indică faptul că o variabilă nu este setată.

```
$a = null; // a este null
```

Obs.: Un aspect important al limbajului PHP este acela că nu acordă atenție asupra tipurilor de date dacă le punem ca și citat, chiar daca este vorba de un întreg sau număr real, PHP face automat conversia de la șir în număr, permițându-i să facă calcule fără să apeleze la funcții dedicate. Acest aspect poate avea consecințe neașteptate.

Când PHP folosește semnul (**+**), încearcă să realizeze operația de adunare, și pentru asta încearcă să convertească șirurile în numere întregi sau reale.

2.3.4 Verificare tipului de date

Pentru verificarea tipului de date folosim funcția **var_dump()**.

```
$integer = 3;
$float = 2.2;
$string = "Acesta este un string";
$boolean = true;
print(var_dump($float));   // float(2.2)
print(var_dump($integer)); // int(3)
print(var_dump($string));  // string(21) "Acesta este un string"
print(var_dump($boolean)); // bool(true)
```

Obs.: În majoritatea cazurilor, PHP convertește automat tipul unei variabile către alt tip de date cu cel mai apropiat context. Acest procedeu este cunoscut ca și "jonglarea tipului". Chiar și așa, uneori este necesar să schimbăm tipul de dată în mod explicit pentru a nu avea surprize la folosirea unui operator.

2.3.5 Verificarea definirii unei variabile

Unul dintre cele mai comune teste este cel de verificare dacă o variabilă este sau nu definită. Simpla utilizare a variabilei ca parametru în funcția **isset()** rezolvă această cerificare (după exemplul care urmează):

```
$integer = 3;
if (isset($integer)) {
    print('Variabila are valoare');
} else {
    print('Variabila nu are valoare');
}
```

2.4 Constantele în PHP

O constantă este un identificator (nume) pentru o valoare. Valoarea nu poate fi schimbată cât timp codul rulează. Un nume valid începe cu o litera sau cu "_" (fără semnul $ înainte de numele constantei).

2.4.1 Crearea unei constante

Pentru a crea o constanta, folosim funcția **define** a cărei sintaxă este următoarea:

define(*nume*, *valoare*, *cazul insensitiv***)**

Descrierea parametrilor**:**

- **nume**: numele constantei;
- **valoare**: valoarea alocată constantei;
- **cazul insensitiv**: Specificarea oricărui nume de constanta trebuie sa fie în cazul insensitiv. Valoarea predefinită pentru acest parametru este **false**.

```
define("GREETING", "Bine ati venit la Atelierele Ilbah");
echo GREETING; // Rezultat: Bine ati venit la Atelierele Ilbah
```

Obs.: O constantă reprezintă o valoare fixă ce nu poate fi schimbată. Toate constantele predefinite sunt scrise cu literă mare. Spre deosebire de variabile, nu pot începe cu '**$**'. De exemplu, constanta pentru pi este **M_PI**. Puteți citi mai multe accesând : Acest link.

Obs.: Spre deosebire de variabile, constantele sunt automat globale cât timp scriptul rulează!

```
define("GREETING", "Bine ati venit la Atelierele Ilbah");
function testGlobal(){
    echo GREETING;
}

testGlobal(); // Rezultat: Bine ati venit la Atelierele Ilbah
```

2.5 Funcții în PHP

O funcție reprezintă o secvență de instrucțiuni identificate printr-un nume. Scopul acestora este de a reduce redundanța (repetiția) codului dintr-un program mai mare. Deși PHP are peste 1000 de funcții predefinite ce pot fi apelate direct din cadrul oricărui program, de multe ori este nevoie ca un dezvoltator să își creeze propriile funcții.

Obs.: Funcțiile nu sunt executate automat împreună cu scriptul PHP în care sunt definite dacă nu sunt apelate explicit.

Structura unei funcții este în general următoarea:

```
function numeFunctie()
{
    // linii de cod conținute de funcție;
}

numeFunctie(); // apelarea funcției
```

Obs.: Numele unei funcții trebuie să înceapă mereu cu literă sau cu "_". Acesta trebuie să fie apropiat de funcționalitatea realizată de funcție.

2.5.1 Funcții cu parametri

Funcțiile pot primi și parametrii ce modifică de obicei comportamentul acestora și respectiv informația generată / afișată. Dacă o funcție primește mai mulți parametri, aceștia vor fi separați prin virgulă.

```
function afiseazaNume($param)
{
    echo "Numele meu este: " . $param . "<br>";
}

afiseazaNume("Ion");        // Numele meu este: Ion
afiseazaNume("Maria");      // Numele meu este: Maria
afiseazaNume("Mircea");     // Numele meu este: Mircea

function afiseazaNumeSiPrenume($prenume, $nume)
{
    echo "Eu sunt: " . $nume . " " . $prenume . "<br>";
}

afiseazaNumeSiPrenume("Ion", "Pop");       // Eu sunt Pop Ion
afiseazaNumeSiPrenume("Maria", "Pop");     // Eu sunt Pop Maria
afiseazaNumeSiPrenume("Mirel", "Radoi");   // Eu sunt Radoi Mirel
```

Obs.: În exemplele anterioare nu am specificat ce tip de date trebuie să fie parametrii funcțiilor. PHP asociază automat tipul de date corespunzător în funcție de valorile primite ca parametru. Începând cu versiunea 7 de PHP s-a introdus și opțiunea de declarare a tipului de date. Împreună cu utilizarea declarației privind acceptarea doar a tipurilor de date specificate, va genera o “eroare fatală” dacă tipurile de date nu corespund.

Obs.: Operatorul de concatenare în PHP este punctul ‘ **.** ‘. Acesta este folosit pentru “alipirea” șirurilor de caractere, după cum ați putut observa în exemplul anterior.

```
<?php

declare(strict_types=1); // declarație de utilizare strictă a tipurilor
de date specificate

function adunare(int $param1, int $param2)
{
    echo ($param1 + $param2);
}

adunare(2, 3); // 5
adunare("2text", 3); // generează o eroare
```

2.5.2 Funcții cu parametri predefiniți

Funcțiile pot avea și parametri cu valori predefinite (**default**). În cazul în care o valoare nu este specificată pentru acești parametri, se va folosi valoarea predefinită.

```
function adunare(int $param1, int $param2 = 3)
{
    echo ($param1 + $param2);
}

adunare(3, 3);          // 6
adunare(2);             // 5
```

2.5.3 Returnarea rezultatului unei funcții

După executarea instrucțiunilor conținute de către o funcție, la final se poate returna și o valoare sau o expresie (vector, șir de caractere, obiect). Acest lucru este util în situațiile în care rezultatul obținut dorim să-l atribuim unei variabile sau expresii.

```
<?php

declare(strict_types=1);

function adunare(int $param1, int $param2)
{
    return $param1 + $param2;
}

$rezultat = adunare(2, 3);
echo $rezultat; // 5
```

Obs.: În cazul în care dorim ca expresia returnată de o funcție să fie de un anumit tip. Se poate declara tipul de dată dorit la definirea funcției prin adăugarea simbolului “ **:** ” urmat de tipul de dată ce dorim a fi returnat de către funcție, așa cum este prezentat mai jos:

```
<?php
declare(strict_types=1); // declarație de utilizare strictă a tipurilor
de date specificate

function adunare(float $param1, float $param2): float
{
    return $param1 + $param2;
}
```

2.6 Teme

1. Realizați o funcție ce primește un parametru și afișează un mesaj de forma: "**Ați introdus valoarea:** " urmat de valoarea parametrului.
2. Realizați o funcție ce primește 2 parametri. Aceasta va afișa rezultatul adunării celor doi parametri. Experimentați cu diferite valori.
3. Realizați o funcție ce primește 3 parametri, dintre care 2 au valori predefinite. Aceasta va afișa rezultatul operației de adunare a celor 3 parametri. Experimentați cu diferite tipuri de date și valori pentru cei 2 parametri predefiniți.
4. Realizați o funcție ce primește 2 parametri și returnează valoarea rezultată prin concatenarea celor 2 parametri. Această valoare va fi asignată unei variabile $rezultat. Se va afișa extern funcției care este tipul variabilei $rezultat.