Laborator 03 - Arrays, Magic Methods

- Obiective:
  - cunoașterea mecanismului din spatele Array-urilor în PHP
  - iteratori peste Array-uri in PHP
  - mecanisme de sortare
  - formatarea și prelucrarea datelor obținute din baza de date
  - înțelegerea modului de funcționare al metodelor magice

Arrays in PHP

Crearea unui Array nou.

Crearea unui `Array` vid:

```
$arr = array();
```

Crearea unui `Array` simplu:

```
$arr = array ("foo", true, 12);
echo $arr[0];
echo $arr[1];
echo $arr[2];

print_r($arr);
```

Observati functia `print_r` folosita pentru afisarea `Array`-urilor.
De asemenea putem observa si urmatoarele:

- Un array poate contine orice tipuri de date (siruri de caractere, boolean, intregi);
- Alocarea se face automat. (La pozitia `0` gasim prima valoare, samd);

In realitate, `Array`-urile in **PHP** sunt implementate ca perechi (sau asocieri) *cheie*"-"*valoare*. O astfel de pereche se reprezinta in PHP astfel:

```
cheie => valoare
```

Obs: Cheile nu pot fi decat intregi sau siruri de caractere.
In exemplu de mai sus, `$arr` contine perechile *cheie-valoare*: `( 0 ⇒ "foo", 1 ⇒ true, 2 ⇒ 12)`. Cum cheile nu au fost declarate explicit (ci doar valorile), acestea sunt generate si incrementate automat de PHP.
Fie urmatorul exemplu:

```
$arr = array (9 => "foo", "bar");
print_r($arr); //va afisa: Array ( [9] => foo [10] => bar )
```

In exemplu de mai sus, valoarea "`foo`" are asociata explicit cheia `9`, insa "`bar`" nu are asociata o cheie explicita. Aceasta se va genera automat.

Mai general, un `Array` poate fi folosit ca:

- vector
- lista
- tabela de dispersie
- stiva
- coada

Modificarea unui array

Modalitatea de adaugare a unui element nou, intr-un `Array` existent:

```
$arr = array("Foo");
$arr [] = "Bar";

print_r($arr); // va afisa: Array ( [0] => Foo [1] => Bar )
```

Modalitatea de adaugare a unui element nou, intr-un `Array` existent, prin specificarea explicita a cheii:

```
$arr = array(1);
$arr ["Foo"] = "Bar";
```

Search

Table of Contents

Laborator 03 - Arrays, Magic Methods
Arrays in PHP
Crearea unui Array nou.
Modificarea unui array
Alte modalitati de parcurgere a Array-urilor (iterare)
Mecanisme de sortare
Utilizari ale Array-urilor
Metode magice in clase
Link-uri utile
Exercitii
Bibliografie

- Repartizare teme
- Catalog PW

Laboratoare

- Laborator 01 - Introducere
- Laborator 02 - BD in PHP
- Laborator 03 - Arrays, Magic Methods
- Laborator 04 - (X)HTML, CSS
- Laborator 05 - Formulare HTML, Persistența Datelor
- Laborator 06 - Securitate I
- Laborator 07 - Securitate II
- Laborator 08 - JavaScript
- Laborator 09 - DOM scripting
- Laborator 10 - AJAX
- Laborator 11 - Web Frameworks

Teme

- Tema 01 - API pentru BD
- Tema 02 - Template System & Controller
- Tema 03 - Generator de formulare
- Tema 04 - Tree Web UI

Login

```
print_r($arr);// va afisa: Array ( [0] => 1 [Foo] => Bar )
```

Stergerea unui element dintr-un `Array`:

```
$arr = array("key" => "value");
unset($arr["key"]);

print_r($arr);// va afisa: Array ( )
```

Dimensiunea unui `array` se poate obtine astfel:

```
$array = array (1, 2, 3);
echo count($array); //va afisa: 3
```

Alte modalitati de parcurgere a Array-urilor (iterare)

In PHP fiecare `Array` are un pointer intern, care permite parcurgerea element cu element a acestuia, in ordinea in care adaugarii elementelor. Aceste functii sunt:

- `current ($array)`: intoarce elementul curent din `Array`;
- `next ($array)`: avanseaza pointer-ul intern pe pozitia urmatoare, si intoarce continutul acesteia;
- `each ($array)`: intoarce perechea (cheie ⇒ valoare) curenta din `Array`, si avanseaza pointer-ul intern pe pozitia urmatoare
- `reset ($array)`: reseteaza pointer-ul curent, la primul element din vector.

Exemplu de folosire (`next` & `current`):

```
$array = array('red', 'blue', 'green', 'yellow');

while($value = next($array)) {
   echo $value, " ";
}// se va afisa, in aceasta ordine: blue green yellow

reset($array);

echo current($array);// va afisa: red
```

Observati faptul ca **primul element din vector a fost sarit**. `Next` **intai** avanseaza la noua pozitie, apoi o afiseaza.

Exemplu de folosire (`each`):

```
$array = array('color' => 'red', 'fruit' => 'apple');

while($pair = each($array)) {
   echo $pair["key"]," ",$pair["value"], ";";
}//se va afisa: color red; fruit apple;
```

Observatii:

- `each` intoarce perechea curenta, tot sub forma unui vector; "`key`" va contine cheia curenta, "`value`" va contine valoarea curenta
- `each` functioneaza diferit fata de `next`; `each` intai intoarce perechea curenta, apoi avanseaza cursorul

Mecanisme de sortare

In PHP, sortarea vectorilor se poate face folosind functia `sort`. Ea primeste ca parametrii:

- vectorul de sortat;
- tipul sortarii, care poate fi:
  - `SORT_REGULAR` - compara elementele normal (fara modificari de tip)
  - `SORT_NUMERIC` - sorteaza numeric elementele
  - `SORT_STRING` - sorteaza elementele ca string-uri (lexicografic)

Exemplu:

```
$array = array('banana', 'apple', 'orange', 'blueberry');
sort($array, SORT_STRING);
print_r($array);
```

Functia `asort` are un comportament asemanator cu `sort`, insa la modificarea listei, pastreaza cheile elementelor, in timpul sortarii. Spre exemplu, `array`-ul:

```
( 0 => "c", 1 => "b" , 2 => "d", 3 => "a")
```

va arata sortat cu `asort` astfel:

```
( 3 => "a", 1 => "b", 0 => "c" , 2 => "d")
```

Sortarea se poate face si dupa criterii particulare, astfel:

- se defineste o **functie-comparator** care primeste ca parametri doua elemente oarecare (de tipul celor din vectorul de sortat), pe care le compara. Functia intoarce 1, 0 sau -1 in functie de rezultatul comparatiei.
- se apeleaza functia `usort`, care primeste ca parametrii vectorul de sortat, si numele functiei-comparator.

Exemplu (cod care sorteaza crescator un vector de vectori):

```
//functie care compara doi vectori, dupa dimensiunea elementelor
function compare ($a, $b) {
   return count($a) > count($b) ? -1 : 1;
}
$array = array( 0 => array('red', 'blue'),
                1 => array('apple', 'banana', 'blueberry'),
                2 => array() );

usort($array, "compare");
print_r($array);
```

Utilizari ale Array-urilor

In cele ce urmeaza, sunt prezentate diverse ipostaze de folosire a `Array`-urilor. Unele mecanisme se pot intersecta.

Array ca si lista

```
$list = array('red', 'blue', 'green');
$elem = current($list);
do {
   echo $elem,",";
}while($elem = next($list));
```

Array ca si colectie

```
$colors = array('red', 'blue', 'green', 'yellow');

foreach ($colors as $color) {
    echo $color;
}
```

Array ca si coada

```
$queue = array("orange", "banana", "apple");
$elem = array_shift($queue);

echo $elem; // va afisa: "orange"
print_r($queue);// va afisa: Array ( [0] => banana [1] => apple )

array_push($queue, "blueberry");
print_r($queue);// va afisa: Array ( [0] => banana [1] => apple [2] => blueberry )

echo array_shift($queue); // va afisa: banana
```

Array ca si stiva

```
$stack = array("orange", "banana");
array_push($stack, "blueberry");

print_r($stack);// va afisa: Array ( [0] => orange [1] => banana [2] => blueberry )

$elem = array_pop($stack);
echo $elem; // va afisa: blueberry
```

Array ca si tabela de dispersie

```
$hashTable = array("key1" => "value1", "key2" => "value2");
foreach ($hashTable as $key => $value) {
    echo $key," ",$value;
}
```

Se observa in exemplul de mai sus, parcurgerea lui `$array`, element cu element. (Unde fiecare element reprezinta o intrare de tip cheie ⇒ valoare).

Metode magice in clase

Clasele in PHP suporta o serie de metode, numite metode magice (magic methods), care au o functionalitate speciala in limbaj, si sunt de obicei apelate automat de interpretorul PHP cand se indeplinesc anumite conditii. Toate metodele magice incep cu doua caractere underscore, si manualul

PHP recomanda sa se evite folosirea numelor de functii care incep in acelasi mod.

Un astfel de exemplu sunt

```
__construct() si __destruct()
```

despre care am invatat in laboratoarele trecute. Aceste metode se apeleaza automat atunci cand un obiect este creat (cu new), repectiv cand un obiect este distrus (la sfarsitul executiei unui script sau in mod explicit undeva in cod).

Alt exemplu ar fi

```
void __set ( string $name ,  mixed $value )

mixed __get ( string $name )

bool __isset ( string $name )

void __unset ( string $name )
```

unde mixed inseamna ca pot fi mai multe tipuri de obiecte, dar nu neaparat toate.

Aceste metode sunt utilizate pentru realizarea supraincarcarii (overloading) membrilor clasei. Ele sunt apelate ori de cate ori se incearca accesul la o variabila membru care nu exista sau la care nu exista acces (declarata private), astfel:

```
__set() se apeleaza atunci cand se incearca scrierea intr-o variabila membru innacesibila
__get() se apeleaza atunci cand se incearca citirea dintr-o variabila membru innacesibila
__isset() este apelata atunci cand se apeleaza isset() sau empty() pe variabile membru innacesibila
__unset() se apeleaza atunci cand se incearca apelul unui unset() pe o variabila membru innacesibil
```

Un exemplu unde utilizarea acestor metode este util este la accesul mai comod la variabile al caror nume se determina la runtime, nu in timpul executiei codului.

Aceste metode nu pot fi folosite decat in context obiect, si nu pot fi folosite in contexte statice.

```
__toString()
```

Aceasta metoda are comportamentul pe care il are si in alte limbaje de programare, si anume este apelat automat la incercarea de transformare in String a unei instante a obiectului (spre exemplu, la un apel echo $obiect)

Alte metode magice ce merita mentionate sunt:

```
mixed __call ( string $name , array $arguments )

mixed __callStatic ( string $name , array $arguments )

mixed __sleep()

mixed __wakeup()
```

call() si callStatic() sunt apelate atunci cand se incearca apelul unei metode a unei clase si clasa nu are definita sau nu este accesibila acea metoda (este declarata private). Prin utilizarea call() si callStatic() se poate implementa overloading pentru metode in PHP, dar pot fi folosite si în alte scopuri.

sleep() si wakeup() se apeleaza automat la serializarea / deserializarea (functiile serialize() și unserialize()) obiectelor, si sunt utilizate pentru eliberarea / re-creearea resurselor neserializabile (spre exemplu, conexiunea la baza de date este o resursa neserializabila).

Link-uri utile

Overloading - PHP.net

Exercitii

Descărcați această arhivă.

- **(1p)** Scrieti o clasa **UserCollection** pentru parcurgerea si afisarea utilizatorilor dintr-un tabel. Trebuie sa folositi un vector de obiecte de tip User. Clasa trebuie sa permita urmatoarele facilitati:
- **(1p)** sortarea crescator/descrescator dupa un camp specificat (**Trebuie folosit un mecanism de sortare vectori**)
- **(2p)** introducerea de restrictii; un set de restrictii poate fi reprezentat de un array (cheie ⇒ valoare), avand urmatoarea semnificatie: *Selecteaza din tabel doar inregistrarile unde coloana "cheie" are valorile "valoare"*. Exemplu: daca array-ul de restrictii este de forma ("email" ⇒ "test@email.com") atunci lista va contine doar utilizatorii care au campul `email` egal cu valoarea "test@email.com";
- informatiile legate de sortare si de restrictii pot fi trimise in constructorul clasei, si retinute ca membri ai clasei;

- **(2p)** Adaugati clasei **UserCollection** metodele:
  - getNextUser(): intoarce un obiect de tip `User`, sau `false` (daca nu mai sunt utilizatori in

colectie) - Hint: current() si next();
- _ _toString (): afiseaza intr-un format intuitiv colectia;

- **(1p)** Scrieti un mecanism prin care parametrii de sortare/restrictii sa poata fi modificati fara a fi necesara re-instantierea clasei `UserCollection`;
- **(3p)** Re-scrieti clasa de mai sus, fara a folosi array-uri pentru retinerea de informatii; care forma de scriere considerati ca este mai eficienta ?
- **Bonus (2p):** Afisati colectia voastra pe pagini. Numarul total de inregistrari se imparte la numarul inregistrarilor pe pagina, si doar inregistrarile aferente unei pagini sunt afisate. (Pentru aceasta, stabiliti o constanta = numarul de Useri afisati pe o pagina si adaugati un membru pagina curenta, care sa afiseze doar utilizatorii de pe acea pagina.) Pentru limitare, folositi cuvantul cheie LIMIT in query-ul vostru.

Bibliografie

[1] http://php.net/manual/en/language.types.array.php