Programare Web

Limbajul PHP

Ciprian Dobre
ciprian.dobre@cs.pub.ro

Obiective

- Obiectivul acestui curs un este o prezentare completă a limbajului PHP ci o introducere.
- Limbajul este simplu, ușor de învățat, semănând cu alte limbaje de programare deja studiate.
- Există o multitudine de cărți și documentații accesibile online din care se pot aprofunda diversele aspecte ale limbajului și folosirii acestuia.
- În cursurile de PHP vom prezenta atât interfața sa cu SGBD-ul MySQL cât și scurte exemple de lucru cu Oracle și ODBC.

Introducere

- PHP a reprezentat inițial o abreviere pentru “Personal Home Pages tools” şi a fost creat în 1994 de Rasmus Lerdorf.
- În primele versiuni (PHP/FI) el conţinea suport doar pentru pachetul de baze de date mSQL (mini SQL).

- Zeev Suraski șiAndi Gutmans au rescris parserul în 1997, formând prima versiune matură a PHP (3), redenumit în ‘PHP: Hypertext Processor’

- PHP poate fi folosit în conjuncţie cu o largă listă de SGBD-uri, atât comerciale cât şi necomerciale (free software) incluzand MySQL, dBase, Oracle, DB2, PostgreSQL, Sybase, InterBase, SQLServer, ODBC, etc.
- În 2004 a fost prezentat PHP 5, având la bază nucleul Zend Engine II
- PHP 6 este la ora actuală în curs de dezvoltare.

Introducere

- Ca pachet, PHP poate fi instalat atât ca un interpretor de sine stătător (deci un CGI separat) cât şi ca un modul pentru serverul de web (Apache).
- Pachetul poate fi instalat de asemenea sub Windows IIS/PWS si Apache.
- Informaţii complete despre instalare şi configurare în diverse variante pot fi găsite în manualul care îl însoţeşte.

Inserare cod

- Introducerea de inserturi scrise în PHP în fişierele HTML poate fi făcută în mai multe moduri (funcţia `echo` are acelaşi efect cu `printf` din limbajul C):

1. <?php echo "Varianta 1 tip XML\n"; ?>
2. <? echo "Varianta 2, cea mai simpla\n"; ?>
3. <script language="php">

   echo "Varianta 3, tip limbaj pentru scripturi";

   </script>;
4. <% echo "varianta 4, tip ASP"; %>

Inserare cod

- Pentru folosirea unora dintre aceste variante interpretorul de PHP trebuie configurat corespunzător (variantele 2 și 4)
- Opțiunile respective se găsesc în fișierul de configurare php.ini

```
; Allow the <? tag. Otherwise, only <?php and <script> tags are recognized.
; NOTE: Using short tags should be avoided when developing applications or
; libraries that are meant for redistribution, or deployment on PHP
; servers which are not under your control, because short tags may not
; be supported on the target server. For portable, redistributable code,
; be sure not to use short tags.
short_open_tag = On

 ; Allow ASP-style <% %> tags.
 asp_tags = Off
```

Descriere

- Insertul poate conţine o singură instrucţiune, ca în exemplele de mai sus, sau mai multe instrucţiuni.
- Formatul instrucţiunilor este liber, putându-se continua pe mai multe linii.
- Ca şi în C fiecare instrucţiune este terminată cu punct şi virgulă.
- Directiva de închidere bloc *php* ține și loc de ;

Exemple

- Obișnuit:

```
<?php
echo 'Hello world';
?>
```

- Fără ; în final de bloc:

```
<?php echo 'Hello world' ?>
```

- Fără directiva de închidere la ultimul bloc (dar cu ; !)

```
<?php echo 'Putem omite directiva in final';
```

Comentarii

- Comentariile sunt ca și in C / Unix shell:

```
<?php
echo 'Test'; // Comentariu pe linie
/* Comentariu pe
    potential mai multe linii */
echo 'Ceva'; # Comentariu pe linie
?>
```

Tipuri și variabile

- PHP este un limbaj interpretat. În consecință nu este necesară declararea variabilelor.
- O variabilă împrumută tipul valorii stocate în ea la momentul respectiv.
- O variabilă își poate schimba tipul pe parcursul execuției scriptului PHP.
- O variabila se creează în momentul în care se depune o valoare în ea.
- Pentru conversia între tipuri (daca e necesară) se pot folosi construcţii de conversie de tip `cast` similare cu cele din C sau funcţia `settype`.

Tipuri

- Există 8 tipuri de date în PHP: 4 tipuri scalare, două compuse și două speciale

1. Boolean
2. Integer
3. Float (include double)
4. String
5. Array
6. Object
7. Resource
8. Null

Variabile

- Numele oricărei variabile este prefixat cu simbolul $.
- Acesta e un marcaj de variabilă și nu face efectiv parte din nume.
- Pentru aflarea tipului unei variabile se poate folosi funcția *gettype* sau var_dump.
- Instrucţiunea de atribuire este identică cu cea din limbajul C.
- Dacă variabila nu este deja definită, ea se crează automat:

```
$nume = "Ion";
$adresa = "Bucuresti";
$sir = "1234";
$numar = (int) $sir; // exemplu de cast
```

Colectarea memoriei

- În PHP memoria este automat colectată – programul cunoaște când o variabilă nu mai este folosită și colectează automat memoria ocupată de aceasta
- Singura excepție: conexiunile cu bazele de date

Tipul Boolean

- O variabilă de tip Boolean poate conține valoarea True sau False.
- În cazul conversiei la Boolean, sunt considerate False (printre altele):
  - Literalul FALSE
  - Valoarea întreagă sau reală (float, double) 0 (zero)
  - Un șir vid
  - Șirul "0"
  - Un array cu 0 elemente
  - Tipul special NULL (incluzând variabilele ne-setate = inexistente)
- Orice altă valoare este considerată True (inclusiv orice resursă)

Tipul întreg

- Specificarea se poate face în bazele 10, 8 și 16:

```
<?php
$a = 1234; // zecimal
$a = -123; // zecimal, negativ
$a = 0123; // octal
$a = 0x1A; // hexazecimal
?>
```

Numere întregi

- Dacă o cifră este incorectă restul cifrelor se ignoră:

```
<?php
$a = 123EU4PLECLA5678; // 123
$a = 012389; // octal 0123
$a = 0x1ASPARAGUS; // hexa 0x1A
?>
```

- Dacă se depășește capacitatea de reprezentare pentru întregi (~2^31), valoarea devine automat `float`.
- La conversia de la float la intreg numărul e rotunjit spre 0.
- Dacă se depășește capacitatea de reprezentare pentru întregi rezultatul este nedefinit (nu se emite nici o atenționare!)
- La conversia de la string la număr se ia prefixul întreg al numărului (ca mai sus):

```
$a = 1 + "3 iezi cucuieti"; // $a devine 4
$a = "3 iezi cucuieti" + 1; // $a devine 4
```

Tipul real (float, double)

- Se pot scrie în formatul uzual sau exponențial:

```
<?php
$a = 1.234;
$b = 1.2e34;
$c = 12E-34;
?>
```

- Valorile limită sunt dependente de platformă, dar uzual numerele sunt până la ~1.8e308 cu o precizie de 14 cifre.
- Conversia de la șir la float se face similar cu cea a întregilor (până la primul caracter care nu face parte dintr-o reprezentare corectă de număr real).

Tipul ARRAY

- Un tablou PHP este o succesiune de perechi (cheie, valoare).
- Li se mai spune și tablouri asociative.
- Un tablou poate fi exploatat în modul clasic (chei pornind de la 0 ca în C) sau ca tablou asociativ (acces prin cheie, cheile putând să nu fie succesive și nici numerice.

Exemple

```
<?php
   $a = array("pw" => "examen",
              4 => "an terminal",
              "succes" =>true);
   echo $a["pw"], $a[4]  // examen an terminal
// $b = array cu 2 dimensiuni
   $b = array("medii" => array(1 => 9.45, 2 => 9.5, 3
              => 8.12, 4 => 9.90, "stat"=>10));
   echo $b["medii"][2]; // 9.5
   echo $b["medii"]["stat"]; // 10
// $c e identic cu $b
   $c = array("medii" => array(1 => 9.45, 9.5, 8.12,
              9.90, "stat"=>10));
   echo $c["medii"][2]; // 9.5
   echo $c["medii"]["stat"]; // 10
?>
```

Tipul Array

- Cheia trebuie să fie scalară (un alt array sau obiect)
- Adăugarea încă unui element cu cheie maximă negativă adaugă o pereche cu cheia 0 (începând cu v4.3.0)
- Cheia TRUE devine 1
- Cheia FALSE devine 0
- Cheia NULL devine șirul vid

Exemplu

```
$regiune = array(-12 => “Oltenia”);
// o variabila de tip array cu cheia
// maxima -12
// Adaugam noi elemente si vom crea noi
// perechi cu chei incepand cu 0:
$regiune[] = "Muntenia"; // elementul 0
$regiune[] = "Moldova"; // elementul 1
```

- În lipsă, cheile pleacă de la 0:

```
$orase = array(“Bucuresti”, “Ploiesti”,
“Campina”) // chei 0, 1, 2
```

Conversii

- La conversia din tipurile întreg, real, string, boolean și resursă în tipul array se crează un tablou cu un singur element cu cheia 0 și valoarea respectivă.
- Dacă se convertește un obiect la array, obținem un array având ca elemente proprietățile obiectului. Mai multe amănunte în documentația PHP.
- Conversia unei valori nule la array duce la un array vid (Atenție: vid nu înseamnă nul!)

Comparații

- 2 tablouri se pot compara astfel:
  - Egalitate: $a == $b adevărat daca au aceleași perechi (cheie, valoare)
  - Identitate: $a === $b adevărat dacă au aceleași perechi (cheie, valoare) în aceeași ordine și cu aceleași tipuri
  - Inegalitate: $a <> $b sau $a != $b. Inversa egalității.
  - Nonidentitate: $a !== $b. Inversa identității.

Comparații

- Se pot afla diferențele dintre 2 tablouri folosind funcția array_dif care returnează valorile dintr-un array care nu se găsesc în al doilea:

```
<?php
    $array1 = array("ion", "vasile", "ion", "elena");
    $array2 = array("vasile", "ion", "mia");
    $rezultat = array_diff($array1, $array2);
    print_r($rezultat);
    // rezultat: [3]=>"elena"
?>
```

Afișează informații despre conținutul unor variabile în format *human-readable.*
Pentru exemplul nostru se afișează:
Array ([3] => elena)

Reuniune

- Două tablouri se pot reuni folosind operatorul +:

```
$c = $a + $b
```

- Rezultatul conține perechile primului array la care se adaugă perechile din al doilea array cu o cheie care nu există în primul.
- Exemplu:

```
<?php
  $array1 = array("ion", "vasile", "ion", "elena");
  $array2 = array("vasile", "ion", "mia");
  $rezultat = $array2 + $array1;
  print_r($rezultat);
?>
```

- Vom obține un tablou cu 4 elemente: elementele din array2 (chei 0, 1, 2) și ultimul element din array 1 (cheia 3):

```
("vasile", "ion", "mia", "elena")
```

Exemplu

- Același exemplu, însă altă adunare:

```
<?php
  $array1 = array("ion", "vasile", "ion", "elena");
  $array2 = array("vasile", "ion", "mia");
  $rezultat = $array1 + $array2;
  print_r($rezultat);
?>
```

- Vom obține un tablou cu 4 elemente: elementele din array1 (chei 0, 1, 2 si 3) și nici un element din array 0 (pentru că toate cheile deja există):

  ("ion", "vasile", "ion", "elena")

Tipul Obiect

- Un obiect se declară prin folosirea operatorului new urmat de un constructor de clasă:

```
$objectName = new ClassName();

$objectName = new ClassName(97.58, 1);
```

- Metodele și proprietățile unui obiect se accesează folosind ->.

```
$Checking->getBalance();

$CheckNumber = 1022;

$Checking->getCheckAmount($CheckNumber);
```

Clase

- Pentru crearea unei clase se folosește cuvântul cheie class:

```
class ClassName {
// membrii și funcții membre
}
```

- Numele claselor poate fi obținut cu get_class:

```
$Checking = new BankAccount();
echo 'The $Checking object is instantiated from the '.
  get_class($Checking) . "class.</p>";
```

Clase externe

- Clasele pot fi declarate în fișiere externe, caz în care se pot folosi funcțiile:
  - include()
  - require()
  - include_once()
  - require_once()

Specificatori de acces

- Specificatorii de acces determină drepturile de acces asupra membrilor unei clase
- În PHP există trei nivele de protecție:
  - public
  - private
  - protected
- Public – oricine are acces la membrii clasei
- Private – se restricționează dreptul de acces asupra membrilor clasei

```
class MyClass {
    private $id = 18;
    public function getId() { return $this->id; }
}
```

Exemplu

```
class Person {

    private $name;

    function __construct($name) {
        $this->name = $name;
    }

    function setName($name) {
        $this->name = $name;
    }

    function getName() {
         return $this->name;
    }
};
```

```
$judy = new Person();

$judy->setName("Judy");

$joe = new Person();

$joe->setName("Joe");

print $judy->getName() . "\n";
//print Judy

print $joe->getName(). "\n";
//print Joe

$judy = new Person("Judy");

$joe = new Person("Joe");

print $judy->getName();

print $joe->getName();
```

Proprietăți statice

```
class MyUniqueIdClass {

    static $idCounter = 0;

    public $uniqueId;

    function __construct() {
        self::$idCounter++;
        $this->uniqueId = self::$idCounter;
    }
}
```

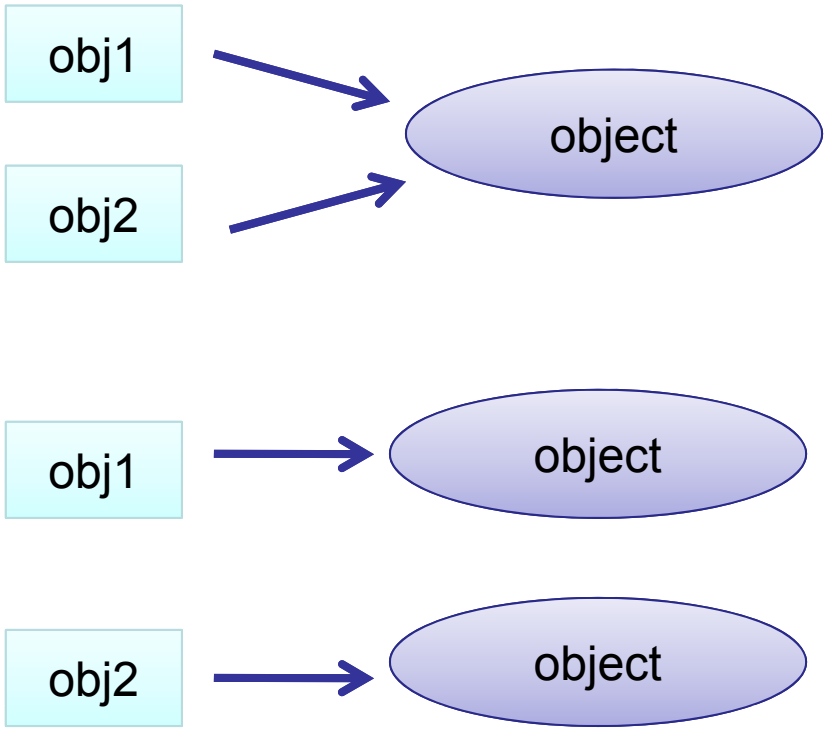
```
$obj1 = new MyUniqueIdClass();

print $obj1->uniqueId . "\n";
//print 1

$obj2 = new MyUniqueIdClass();

print $obj2->uniqueId . "\n";
//print 2
```

Clonarea obiectelor




```
class MyClass {
   public $var = 1;
}
```

```
$obj1 = new MyClass();
$obj2 = $obj1;
$obj2->var = 2;
print $obj1->var; //print 2
```

```
$obj1 = new MyClass();
$obj2 = clone $obj1;
$obj2->var = 2;
print $obj1->var; //print 1
```

Polimorfism

- Mecanism de moștenire a unei singure clase
  - Similar Java
- Însă se pot implementa mai multe interfețe

```
class Child extends Parent {
…
}
```

```
class A implements B, C, ... {
…
}
```

```
interface I1 extends I2, I3, ... {
…
}
```

Tipul Obiect

- … mai multe detalii la orele de laborator.
- Dacă o valoare de alt tip este convertită la tipul obiect, obținem o instanță a clasei stdClass.
- Dacă se convertește la obiect o valoare nulă, noua instanță va fi vidă.
- Dacă un tablou se convertește la obiect cheile devin proprietăți.

Tipul Resursă

- Tipul resursă este un tip special, variabilele de acest tip conțin o referință către o resursă externa.
- Felul resursei se poate obține cu funcția `get_resource_type` (ex: mysql link, file, domxml document, etc.).
- Conversia la tipul resursă nu are sens (din definiția tipului)
- O resursă care nu mai este referită este detectată automat de 'garbage collector' și eliberată (deci nu e necesară eliberarea manuală).

Tipul NULL

- Tipul NULL are o singură valoare, NULL.
- O variabilă nula nu conține nici o valoare.
- O variabilă este considerată nulă dacă:
  1. I-a fost asignată valoarea NULL
  2. Nu i-a fost asignată încă nici o valoare (deci eventual ea nu există).
  3. A fost dealocată cu funcția `unset($variabila)`
- O variabilă se poate testa dacă e nulă sau nu cu funcția `is_null($variabila)`.
- O variabilă se poate testa dacă există sau nu (cazurile 2 și 3 de mai sus) cu funcția `isset($variabila, ...)`
- O variabilă se poate testa dacă este goală cu `empty($variabila)`. Sunt considerate goale variabilele care conțin echivalentul lui FALSE (inclusiv care conțin valoarea NULL).

Variabile

- Numele variabilelor este prefixat cu $ (marcaj de variabila).
- Numele este case-senzitiv (literele mari sunt considerate diferite de cele mici).
- Un nume corect PHP începe cu literă sau underscore și continuă cu litere, cifre și underscore.
- Variabilele pot conține referințe către alte variabile (adresa se prelevează cu &, ca în C). Exemplu:

Variabile

```
<?php
$unu = 'Ceva';
$doi = &$unu; // $doi e o referinta la $unu.
$doi = "Altceva"; // Modificam $doi
echo $unu; // Ambele contin acelasi
echo $doi; // sir
?>
```

- Nu se poate preleva cu & adresa unei expresii (doar a unei variabile).
- PHP inițializează variabilele cu valori implicite dar nu este bine să ne bazăm pe aceste valori (0 pentru numere, False pentru boolean, etc).

Variabile predefinite

- Există un număr mare de variabile predefinite (de sistem) pe care scriptul le poate folosi.
- Cele mai utilizate sunt:
  - $_GET, $_POST, $_COOKIES, $_REQUEST – conțin valorile transmise scriptului cu metodele GET, POST, cookie sau reuniunea lor
  - $_SESSION – conține variabile care se pot folosi de o succesiune de execuții de scripturi care formează o sesiune de lucru
  - $GLOBALS – conține toate variabilele globale ale scriptului
- O descriere a acestor variabile și a altora din aceeași categorie se găsește în documentația PHP

Domeniul (scope)

- Variabilele sunt cunoscute în contextul în care au fost create.
- Cele definite în afara oricărei funcții sunt similare variabilelor globale din C.
- Cele definite în funcții (prin asignare) sunt locale acelei functii.
- Spre deosebire de limbajul C o variabilă globală nu este cunoscută în interiorul unei funcții decât dacă este declarată cu *global* în acea funcție.

Domeniul (scope)

- Exemplul 1:

```
$a = 1; /* variabila globala */
function Ecou() {
    $a = 2; /* se asigneaza o valoare variabilei locale */
    echo $a; /* tiparire variabila locala */
}
Ecou();
echo $a; /* tiparire variabila globala */
```

- Rezultat 21

Domeniul (scope)

- Exemplul 2:

```
$a = 1; /* variabila globala */
function Ecou() {
    global $a
    $a = 2;/* se asigneaza o valoare variabilei globale */
    echo $a;/* tiparire variabila globala */
}
Ecou();
echo $a; /* tiparire variabila globala */
```

- Rezultat 22

Domeniul (scope)

- Un alt mod de a defini variabile globale cunoscute și în interiorul funcțiilor este folosirea tabloului asociativ predefinit $GLOBALS, având ca indici numele variabilelor globale:
- Exemplu:

```
$a = 1; /* variabila globala */
function Ecou() {
    $GLOBALS["a"] = 2; /* se asigneaza o valoare
                          variabilei globale */
    echo $GLOBALS["a"];/* tiparire variabila globala */
}
Ecou();
echo $a; /* tiparire variabila globala */
```

- Rezultat 22

Variabile statice

- Ca și în C se pot defini variabile locale funcțiilor dar care își păstrează valoarea de la un apel la altul.
- Aceste variabile se numesc statice.
- Variabilele statice pot fi inițializate cu o valoare care apoi se modifică și este păstrată pentru apelurile viitoare:

```
function Increment() {
    static $a = 0;
    echo $a;
    $a++;
}
```

- Rezultatul apelului repetat al acestei funcții va fi afișarea numerelor 0, 1, 2, . . .

Macrosubstituție

- Numele unei variabile se poate găsi în altă variabila.
- Acest procedeu, numit macrosubstituție, este întâlnit și în alte limbaje, cum este Xbase (dBase, Fox, Clipper).
- Instrucțiunile:

```
$a = "Limbajul";
$$a = " PHP";
```

- definesc două variabile: $a cu valoarea “Limbajul” și $Limbajul cu valoarea “ PHP”.
- În acest caz instrucțiunea:

```
echo "$a ${$a}";
```

va afișa Limbajul PHP

Variabile externe

- În această categorie intră variabilele corespunzătoare simbolurilor primite de la un formular și variabilele de mediu setate de serverul de web.
- Să presupunem că avem următorul formular:

```
<form action="actiune.php" method="post">
    Nume: <input type="text" name="nume"><br>
    Localitate: <input type="text" name="adresa[localitate]"><br>
    Strada: <input type="text" name="adresa[strada]"><br>
    Numar: <input type="text" name="adresa[numar]"><br>
    Optiuni: <br>
    <select multiple name="so[]">
      <option value="Windows 95">Windows 95
      <option value="Windows XP">Windows XP
      <option value="Windows Vista">Windows Vista
      <option value="Linux">Linux
    </select>
    <input type="submit">
</form>
```

Variabile externe

- Scriptul *actiune.php* care tratează acest formular poate primi variabilele:
  - *$nume*, variabilă simplă
  - *$adresa*, un tablou asociativ cu trei elemente
  - *$so*, un tablou având atâtea elemente câte selecții s-au făcut în meniul vertical din formular.
- Acest lucru se întâmplă însă doar dacă opțiunea de configurare a PHP *register_globals* e setată pe *On* (implicit ea e însă Off, fiind potențial o breșă de securitate).
- În mod normal valorile celor 3 variabile se găsesc în $_POST și $_REQUEST.

Constante

- Constantele se definesc similar cu limbajul C, cu define:

```
<?php
    // Constante valide
    define("MATERIE", "Programare Web");
    define("_EVAL_UARE", "Examen");
    define("NOTA10", "10");
    // Nume invalid
    define("2PAC", "Cantaret");
    // Asa arata constantele PHP,
    // e bine sa nu avem si noi la fel
    define("__NOTA__", "10");
?>
```

Constante

- Spre deosebire de variabile:
  - Constantele nu au un nume care începe cu $
  - Constantele pot fi definite doar cu `define()` nu prin atribuire
  - Constantele nu au domeniu de valabilitate ca variabilele (se pot folosi și în funcții de exemplu).
  - Nu pot să-și schimbe valoarea și nu pot fi dealocate (`unset`)
  - Constantele pot conține doar valori scalare (boolean, intreg, real sau șir)

Constante predefinite

- PHP pune la dispoziție și o serie de constante predefinite.
- Acestea au forma __Nume__
- Printre ele sunt:
  - __LINE__ numărul liniei curente în sursa PHP
  - __FILE__ calea și numele complet al fișierului sursă PHP
  - __DIR__ directorul acelui fișier
  - __FUNCTION__ numele funcției curente (doar cu litere mici în PHP4)
  - __CLASS__ numele clasei (doar cu litere mici în PHP4)
  - __METHOD__ numele metodei din clasă (doar PHP5)

Expresii

- Expresiile în PHP sunt similare celor din limbajul C
- Se pot folosi construcții de tipurile:
  - $a++, ++$a
  - $a--, --$a
  - $a += 3; (echivalentă cu $a = $a + 3;), în loc de + putând fi orice operator valid pentru operația respectivă
  - Atribuiri multiple, ca de exemplu:

    $a = $b = ++$c; sau

    $a = $b += 10;

Atribuirile întorc o valoare

- Ca și în limbajul C atribuirile întorc o valoare:

```
if ($con = mysql_connect(...)) ...
```

- Ca și în limbajul C o expresie logică e evaluată doar până în momentul în care valoarea sa este certă:

```
mysql_connect(...) or die('Conexiune esuata');
```

Operatori

- Aritmetici: +, -, *, /, % (modul)
- Logici: ==, ===, !=, !==, <, >, <=, >=
- Conectori logici: and, &&, or, ||, ! (negare), xor (sau exclusiv)
- Operatori pe șiruri: . (concatenare)
- Operatori pe biți: &, |, ~ (inversare biți)

Structuri de control

- Există o serie de structuri de control care sunt similare celor din limbajul C.
- Vom avea ca și acolo decizii, cicluri, alegere
- Se pot defini ca și în C funcții (nu există decât funcții, nu și proceduri ca în Pascal).

Decizia

- În PHP aceste instrucțiuni sunt asemănătoare ca sintaxa cu cele similare din limbajul C.
- Sintaxa:

```
if (conditie_1)
   { instructiuni_1 }
elseif (conditie_2)
   { instructiuni_2 }
. . . . . .
else { instructiuni_N }
```

- elseif și else sunt opționale (similar cu limbajul C).

Decizia

- Exemplu:

```
if ($a > $b)
  { print "a este mai mare ca b";}
elseif ($a == $b)
  { print "a este egal cu b";}
else
  { print "a este mai mic decat b";}
```

Ciclul WHILE

- Sintaxa:

```
while ( conditie )
{ instructiuni }
```

- Exemplu:

```
$i = 10;
while ($i >= 0)
{ print $i--; }
```

Ciclul DO

- Sintaxa:

```
do
{ instructiuni }
while ( conditie );
```

- Exemplu:

```
$i = 10;
do
{ print $i--;}
while ($i>0);
```

Ciclul FOR

- Sintaxa:

```
FOR (expr1; expr2; expr3)
    instrucțiune
```

- Execuția unui astfel de ciclu se face astfel:
  - Se evaluează expresia expr1
  - Cât timp expresia expr2 are valoarea adevărat se repetă operațiile:
    - Se executa instrucțiunea (instrucțiune)
    - Se evaluează expresia expr3
- Exemplu:

```
for ($i = 1; $i <=10; $i++)
{ print $i;}
```

- Efectul va fi afișarea valorilor de la 1 la 10.

FOREACH

- Sintaxa (2 variante):

```
foreach (expr_array as $valoare)
    statement
foreach (expr_array as $cheie => $valoare)
    statement
```

- Se folosește pentru parcurgerea unui tablou (ciclu după elementele unui tablou)

Exemplu

```
<?php
// tiparirea unui tablou
$arr = array("one", "two", "three");
foreach ($arr as $val) {
   echo "Value: $val<br />\n";
}
foreach ($arr as $k => $val) {
   echo "Cheie: $k; Val: $val<br />\n";
}
?>
```

Break și Continue

- Aceste instrucțiuni se folosesc pentru a ieși dintr-un ciclu, respectiv pentru a se trece necondiționat la un nou pas al ciclului chiar dacă pasul curent nu s-a terminat.
- Exemplu:

Tipărirea numerelor impare dintre 1 și 10

```
for ($i = 1;;$i++) {
    if ($i > 10) { break; }
    if ($i % 2) { continue; }
    print $i;
}
```

Alegerea (Switch)

- Sintaxa:

```
switch (expr)
{
case val1:
   instructiuni
case val2:
   instructiuni
. . . . . .
default:
   instructiuni
}
```

Alegerea (Switch)

- Efectul este următorul:
  - Se evaluează expresia `expr`
  - Se parcurg etichetele case (`val1`, `val2`, …) una după alta. În cazul în care se găsește o egalitate, se executa instrucțiunile de la acea eticheta până la prima instrucțiune `break` sau până se sfârșește întregul `switch`.
  - Daca nu exista nici o egalitate se execută instrucțiunile de la `default`.

Alegerea (Switch)

- Exemplu:

```
switch ($i)
{
case 0:
    print "i egal cu 0";
    break;
case 1:
    print "i egal cu 1";
    break;
case 2:
    print "i egal cu 2";
    break;
default:
    print "i nu este egal cu 0, 1 sau 2";
}
```

- De remarcat că dacă instrucțiunile **break** ar lipsi, în cazul în care **$i** este egal cu 0 se tipăresc toate cele patru mesaje iar în cazul în care este egal cu 1 doar ultimele trei.

Funcții

- Programele PHP pot conține funcții definite de utilizator, inclusiv functii recursive.
- Sintaxa definiției unei funcții este următoarea:

```
function nume_functie
(lista_parametri)
{
instructiuni
}
```

- Dacă se dorește ca funcția să întoarcă o valoare, se folosește instrucțiunea:

```
return expresie;
```

Funcții

- Exemplu:

```
function la_patrat ($numar) {
    return $numar * $numar;
}
echo la_patrat(10);
```

Funcții

- Folosirea unei funcții se poate face doar după definiția acesteia.
- Parametri sunt transmiși prin valoare.
- Dacă se dorește transmiterea prin referință a unui argument, se poate folosi construcția &variabila:

```
function la_patrat (&$numar) {
   $numar = $numar * $numar;
}
$a = 10;
la_patrat($a);
echo $a; // tipareste 100
```

Funcții

- Se poate transmite prin valoare adresa sa:

```
function la_patrat ($numar) {
   $numar = $numar * $numar;
}
$a = 10;
la_patrat($a);
echo $a; // tipareste 10
la_patrat(&$a);
echo $a; // tipareste 100
```

Funcții

- La definirea unei funcții se pot asigna și valori implicite pentru argumente.
- În cazul în care acestea lipsesc la apel sunt luate implicit valorile din definiție:

```
function la_patrat ($numar = 4) {
   return $numar * $numar;
}
echo la_patrat(10); // tipareste 100
echo la_patrat(); // tipareste 16
```

Funcții

- Astfel de argumente trebuie să fie ultimele din listă.
- De exemplu secvența:

```
function inmultire ($numar1 = 4, $numar2) {
    return $numar1 * $numar2;
}
echo inmultire(10);
```

va semnala o eroare deoarece automat valoarea 10 va fi asignată primului argument.

Funcții PHP

- PHP pune la dispoziție un număr foarte mare de funcții, atât de uz general cât și funcții specifice accesului la diverse sisteme de gestiune a bazelor de date.
- În continuare sunt prezentate o parte dintre acestea…

Funcții ARRAY

array array(*lista valori);*

- Creează un *array* conținând valorile din listă.

  Pentru un *array bidimensional* se poate folosi operatorul => pentru asocierea celor doi indici.

- Exemplu:

```
$note = array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
$calificative = array("S"=>"Satisfacator",
  "B"=>"Bine","F"=>"Foarte bine");
```

- În primul caz s-a definit un *array* cu o singură dimensiune iar în cel de-al doilea unul cu două dimensiuni.

Funcții ARRAY

void list(lista de valori);

- Asignează o lista de variabile ca și când ar fi un *array*.
- Exemplu:

```
$rezultat = mysql_query("select nume_s,
  grupa from studenti");
while (list($n, $g) =
  mysql_fetch_row($rezultat))
{ print( "Nume: $n<br>Grupa: $g<br>\n"); }
```

Funcții ARRAY

int count(variabila);

- Întoarce numărul de elemente din variabilă.
- Dacă este un *array* acest număr poate fi mai mare ca 1. Dacă variabila nu este setată întoarce 0.

array each(array array);

- Întoarce următoarea pereche (index, valoare) dintr-un *array* sub forma unui tablou având indicii 0 si 1 sau *key* și *value*.

Funcții ARRAY

- De notat că fiecare variabilă de tip *array* are asociat un pointer intern care arată către unul dintre elementele sale iar funcția *each*() se poate folosi pentru parcurgerea sa.
- Exemplu:

```
$calificative = array ("S"=>"Satisfacator",
  "B"=>"Bine","F"=>"Foarte bine");
list($c, $d) = each($calificative);
```

- Atunci:
  - $c are valoarea "S"
  - $d are valoarea "Satisfacator"

Funcții ARRAY

mixed next(array tablou);

- Întoarce următorul element al unui *array* sau *false* dacă nu mai sunt elemente, avansând deci pointerul intern asociat tabloului.
- De notat că dacă un tablou are elemente nule și pentru acestea valoarea întoarsă va fi *false*. De aceea, pentru parcurgerea unui tablou se recomandă folosirea lui each().

Funcții ARRAY

mixed prev(array tablou);

- Întoarce precedentul element al unui *array* sau *false* dacă nu mai sunt elemente, decrementând pointerul intern asociat tabloului.
- Aceeași observație ca la *next* (folosire *each*)

mixed reset(array tablou);

- Setează pointerul intern asociat tabloului la primul element al acestuia.
- Întoarce valoarea acestui element.

int sizeof(array tablou);

- Întoarce numărul de elemente ale unui tablou. Este analog cu count().

Funcții de informare/setare

int error_reporting(int [level]);

- Setează nivelul de erori care sunt raportate de PHP. Valorile respective pot fi cumulate prin adunare în cazul în care se doresc setate simultan mai multe tipuri de raportări.

string getenv(string variabila);

- Întoarce valoarea unei variabile de mediu sau false în caz de eroare.

```
$ip = getenv("REMOTE_ADDR"); // prelevare adresa IP a clientului
```

Funcții de informare/setare

void putenv(string setare);

- Creează o nouă variabilă de mediu. Exemplu:

```
putenv("USER = $user");
```

int phpinfo(void);

- Întoarce o suită de informații privind PHP: opțiuni de compilare, versiune, informații despre serverul de web, variabile de mediu, versiunea sistemului de operare, etc.

string phpversion(void);

- Întoarce sub forma unui sir numărul versiunii PHP folosite.

Alte funcții

void eval(string sir);

- Șirul de caractere (care trebuie sa conțină expresii valide PHP) este evaluat.
- Exemplu:

```
$nume = 'Ion';
$str = 'Ma numesc $nume<br>';
echo $str;
eval( "\$str = \"$str\";" );
echo $str;
```

- Va avea ca efect tipărirea mesajelor:

```
Ma numesc $nume
Ma numesc Ion
```

Alte funcții

void die(string mesaj);

- Afișează un mesaj și termină execuția scriptului.
- Exemplu:

```
$nume_fisier = '/usr/local/date.txt';
$f = fopen($nume_fisier, 'r') or die "Nu se poate
   deschide fisierul $filename";
```

void exit(void);

- Termină imediat execuția scriptului.

Alte funcții

```
void sleep(int secunde);
```

- Întârzie execuția un număr de secunde.

```
echo(string arg1, string [argn]...);
```

sau

```
echo string arg1, string [argn]...;
```

- Evaluează și afișează parametrii primiți ca argument.

Alte funcții

```
print(string arg);
```

- Afișează argumentul.

```
int printf(string format, mixed [argumente]...);
```

- Afișează argumentele formatate conform șirului format (asemănător cu limbajul C).

```
int strcmp(string str1, string str2);
```

- Compara doua șiruri. Întoarce o valoare < 0, egală cu 0 sau >0 după cum str1 este mai mic, egal sau mai mare decât str2. Literele mari și mici sunt considerate diferite.

```
int strlen(string sir);
```

- Întoarce lungimea unui șir de caractere.

Alte funcții

string substr(string sir, int start, int [lung]);

- Întoarce un subșir conținând caracterele începând cu cel dat de start si având lungimea lung. Dacă start este negativ, numărătoarea pentru stabilirea caracterului inițial este făcută de la sfârșitul șirului. Dacă lung este negativ ultimul caracter al subșirului este cel aflat la distanța lung de sfârșitul șirului. Întoarce întotdeauna cel puțin un
- caracter, cel dat de start.
- Exemple:

```
$rest = substr("abcdef", 1); // întoarce "bcdef"
$rest = substr("abcdef", 1, 3); // întoarce "bcd"
$rest = substr("abcdef", -1); // întoarce "f"
$rest = substr("abcdef", -2); // întoarce "ef"
$rest = substr("abcdef", -3, 1); // întoarce "d"
$rest = substr("abcdef", 1, -1); // întoarce "bcde"
```

Alte funcții

```
int is_array(mixed var);
```

- Întoarce *true* dacă variabila este un *array* si *false* altfel.

```
int is_double(mixed var);
int is_float(mixed var);
int is_real(mixed var);
```

- Întorc *true* dacă variabila este un număr real și *false* altfel.

```
int is_int(mixed var);
int is_integer(mixed var);
int is_long(mixed var);
```

- Întorc *true* dacă variabila este un număr întreg și *false* altfel.

Alte funcții

int is_object(mixed var);

- Întoarce *true* dacă variabila este un obiect și *false* altfel.

int is_string(mixed var);

- Întoarce *true* dacă variabila este un șir de caractere și *false* altfel.

int isset(mixed var);

- Întoarce *true* dacă variabila există și *false* altfel.

int unset(mixed var);

- Elimină o variabilă.

string strval(mixed var);

- Convertește la șir de caractere o variabilă scalară.

Bibliografie

- Documentația PHP: http://www.php.net/docs.php
- Câteva cărți disponibile online:
  - Sams - Teach Yourself PHP in 10 Minutes(2005) http://www.net130.com/CMS/Pub/book/book_web/book_web_php/2005_10_19_70383.htm
  - Object Oriented PHP Concepts Techniques and Code (și altele), la adresa: http://cid-846ffdcf0d3320d8.skydrive.live.com/browse.aspx/eBook