Pagini WEB cu PHP 4

Mihai Scorțaru, Claudiu Soroiu

În cadrul acestui articol vom începe să prezentăm modul în care putem beneficia de avantajele oferite de PHP în crearea paginilor Web. Vom începe cu descrierea modalității de inserare a codului PHP în documentele folosite pentru generarea paginilor Web și vom descrie apoi modul de utilizare al variabilelor în PHP.

Inserarea codului PHP

Interpretorul *PHP* parcurge documentul accesat până în momentul în care întâlnește un marcaj de deschidere care indică faptul că textul care urmează trebuie interpretat ca fiind cod *PHP*.

În continuare, textul este interpretat ca fiind cod *PHP* până în momentul în care este întâlnit marcajul de închidere.

Întreg textul care nu este interpretat ca fiind cod *PHP* este furnizat la ieșire în forma în care a fost primit ca intrare de către interpretor.

Există mai multe marcaje care indică începerea unei secvențe de cod *PHP*, dar doar două dintre ele sunt folosite de obicei.

Dacă dorim ca documentul să respecte specificațiile *XML*, atunci singura posibilitate de inserare a cosului *PHP* este folosirea unei secvențe de tipul:

```
<?php
   //cod PHP
?>
```

Cea de-a doua posibilitate este utilizarea marcajului <**SCRIPT**> într-o manieră asemănătoare celei folosite pentru includerea de *script*-uri *JavaScript*. Sintaxa este următoarea:

```
<SCRIPT language = "php">
   //cod PHP
</SCRIPT>
```

Exsită alte două posibilități, dar acestea sunt folosite rar, în situații particulare.

Este permisă "ieșirea" și "intrarea" în "modul *PHP*" de oricâte ori este necesar.

De exemplu, următoarea secvență este interpretată corect:

```
<?php
  if ($expresie){
    ?>
    <B>Adevărat!</B>
    <?php
  }
  else{
    ?>
    <B>Fals!</B>
  <?php
  }
?>
```

Secvența este corectă deoarece, după întâlnirea unui marcaj de închidere (care indică terminarea unei secvențe de cod *PHP*), întregul text întâlnit până la următorul marcaj de deschidere (care indică începerea unei secvențe de cod *PHP*) este furnizat la ieșire nemodificat.

Secvența de mai sus ar putea părea ciudată, dar în unele cazuri (când se lucrează cu texte de dimensiuni mari) această variantă este mai eficientă decât furnizarea textului ca ieșire folosind comenzi de tip **echo**.

Instrucțiuni PHP

Pentru ca instrucțiunile *PHP* să fie interpretate corect, acestea trebuie separate prin caracterul `';'`. Marcajul de închidere care semnalează terminarea secvenței de cod *PHP* inserează, implicit, un caracter `';'` după ultima instrucțiune *PHP*. Așadar, după ultima instrucțiune a fiecărui bloc de instrucțiuni *PHP*, acest caracter poate lipsi.

Comentarii

În *PHP*, comentariile pot fi inserate folosind sintaxele din *C*, *C++* și *shell*-urile *Unix*.

Apariția secvenței de caractere `'//'` sau a caracterului `'#'` indică faptul că restul liniei reprezintă un comentariu. Pentru a insera comentarii pe mai multe linii, acestea tre-

buie delimitate de secvența '/*' la început și de secvența '*/' la sfârșit.

Trebuie observat faptul că, dacă după '//' sau apare '#' un marcaj care indică încheierea secvenței de cod *PHP*, atunci acesta este luat în considerare și textul care urmează nu mai este considerat comentariu.

Variabile în PHP

În *PHP*, o variabilă este reprezentată de semnul '$', urmat de numele variabilei. La fel ca în limbajele *C/C++* sau *Java*, pentru denumirea variabilelor se face distincție între literele mici și literele mari.

Numele unei variabile poate începe cu o liniuță de subliniere ('_') sau o literă. Restul caracterelor care formează numele variabilei pot fi litere, cifre sau liniuțe de subliniere. În *PHP*, sunt considerate litere toate caracterele cuprinse între 'a' și 'z', cele cuprinse între 'A' și 'Z', precum și cele care au codul *ASCII* cuprins între 127 și 255.

Începând cu versiunea 4, au fost introduse *referințele*; astfel, o variabilă poate referi o alta, astfel încât modificarea valorii uneia duce la modificarea automată a valorii celeilalte. O referință la o variabilă este reprezentată de caracterul '&'. Iată un scurt exemplu:

```
<HTML>
  <HEAD>
    <TITLE>
      Referințe
    </TITLE>
  </HEAD>
  <BODY>
    <?php
      $a = "Gazeta de Informatică";
      $b = &$a;
      $b = "GInfo";
      echo $a;
      echo "<BR>";
      echo $b;
    ?>
  </BODY>
</HTML>
```

Figura 1

După cum se poate observa în fereastra din figura 1, modificarea valorii variabilei $b a dus la modificarea automată a valorii variabilei $a.

Variabile predefinite

Există o mulțime de variabile care sunt deja definite și care pot fi utilizate în diferite scopuri. În cele ce urmează vom descrie câteva dintre acestea.

Prin intermediul vectorului $GLOBALS pot fi accesate toate variabilele globale care sunt accesibile *script*-ului *PHP* curent; acest vector este indexat chiar prin numele variabilelor globale.

Vectorul $SERVER conține o serie de variabile ale căror valori sunt setate de *server*-ul *Web*; majoritatea valorilor variabilelor din acest vector depind de mediul de execuție al *script*-ului curent.

Vectorii $_GET și $_POST conțin variabile primite de *script* prin intermediul unor transferuri care folosesc metodele *HTTP* get, respectiv post. De exemplu, prin intermediul acestor vectori pot fi accesate valorile câmpurilor dintr-un formular care a fost completat și transmis folosind una dintre cele două metode.

Vectorul $_COOKIE conține valorile variabilelor care conțin informații referitoare la *cookie*-urile păstrate pe calculatorul utilizatorului care accesează pagina *Web*.

Vectorul $_FILES conține variabile primite de *script* prin intermediul încărcărilor de fișiere prin metoda post.

Vectorul $_ENV conține variabile disponibile prin intermediul mediului în care este executat.

Vectorul $_REQUEST conține variabile disponibile prin intermediul oricărui tip de mecanism cu ajutorul căruia utilizatorul poate introduce date.

Vectorul $_SESSION conține variabilele care corespund sesiunii curente a *script*-ului.

Am dat doar definițiile acestor vectori, chiar dacă acestea nu oferă informații suficiente pentru a înțelege modul lor de utilizare. Pe măsură ce vom avea nevoie de astfel de informații vom prezenta detalii referitoare la modul de utilizare.

Domeniul de vizibilitate al variabilelor

În *PHP* o variabilă poate fi accesată doar în contextul în care a fost definită. De exemplu, o variabilă definită în interiorul unei funcții nu va putea fi accesată decât de instrucțiunile din interiorul acelei funcții. Mai mult, în mod implicit, o variabilă definită în exteriorul unei funcții nu va putea fi accesată în interiorul funcției chiar dacă ea nu este redefinită în cadrul funcției.

Pentru ca o funcție să aibă acces la o variabilă definită în afara ei, variabila trebuie redeclarată ca variabilă globală în interiorul funcției.

Să considerăm următorul exemplu:

```
<?php
  $a = 1;
  function Test(){
    echo $a;
  }
  Test();
?>
```

Nu se va afișa valoarea 1 deoarece instrucțiunea **echo** $a se referă la variabila locală $a care nu este definită, deci nu are nici o valoare.

Vom prezenta acum o versiune în care instrucțiunea **echo** $a va accesa variabila $a definită în afara funcției Test():

```
<?php
  $a = 1;
```

```
  function Test(){
    global $a;
    echo $a;
  }
  Test();
?>
```

O a doua posibilitate de a accesa o variabilă definită în afara funcției este folosirea vectorului `$GLOBALS`:

```
<?php
  $a = 1;
  function Test(){
    echo $GLOBALS["a"];
  }
  Test();
?>
```

O altă facilitate oferită de *PHP* este folosirea variabilelor *statice*. Acestea își păstrează valoarea dacă se iese din contextul în care a fost definită variabila și, ulterior, se revine în acel context. De exemplu, o variabilă statică definită în interiorul unei funcții își păstrează valoarea de la un apel la altul al funcției. Vom arăta acum modul în care pot fi utilizate variabilele statice:

```
<HTML>
  <HEAD>
    <TITLE>
      Variabile statice
    </TITLE>
  </HEAD>
  <BODY>
    <?php
      function Test(){
        static $a = 0;
        echo $a;
        echo "<BR>";
        $a = $a + 1;
      }
      Test();
      Test();
      Test();
    ?>
  </BODY>
</HTML>
```

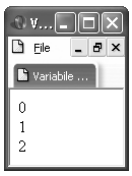

Figura 2

După cum se poate observa în figura 2, la primul apel al funcției `Test()` variabila `$a` are valoarea `0`, la al doilea apel are valoarea `1`, iar la al treilea are valoarea `2`.

Variabile cu nume variabil

Uneori este convenabil să folosim variabile al căror nume poate fi modificat în mod dinamic. Printr-o secvență de tipul `<?php $a = "Hello"; ?>` am creat o variabilă `$a` a cărei valoare este `Hello`. Acum, prin intermediul unei secvențe de tipul `<?php $$a = "World!"; ?>` se creează o variabilă `$Hello` a cărei valoare este `World!`.

În acest moment secvențele `<?php echo "$a ${$a}"; ?>` și `<?php echo "$a $Hello"; ?>` au exact același efect. Codul complet este prezentat în continuare, iar efectul este ilustrat în figura 3.

```
<HTML>
  <HEAD>
    <TITLE>Variabile cu nume variabil</TITLE>
  </HEAD>
  <BODY>
    <?php
      $a = "Hello";
      $$a = "World!";
      echo "$a ${$a}";
      echo "<BR>";
      echo "$a $Hello";
    ?>
  </BODY>
</HTML>
```

Figura 3

Variabile corespunzătoare formularelor

Dacă utilizatorul completează un formular, iar "*feedback*"-ul trebuie furnizat de un *script PHP*, atunci informațiile din formular devin accesibile acestui *script*.

Pentru a indica faptul că datele din formular trebuie procesate de un anumit script *PHP*, atunci numele fișierului care conține *script*-ul trebuie furnizat ca valoarea pentru atributul **action** al marcajului **<FORM>**. Presupunem că avem un formular în care trebuie introdus un nume de utilizator și o parolă, iar aceste date trebuie manipulate de *script*-ul *PHP* stocat în fișierul **form.php**. În aceste condiții, documentul *HTML* care afișează formularul este asemănător cu cel prezentat în continuare (efectul poate fi vizualizat în figura 4):

```
<HTML>
  <HEAD>
    <TITLE>Un formular simplu</TITLE>
  </HEAD>
  <BODY>
    <FORM action = 'form.php' method = post>
      <DIV align = center>
        Numele: 
        <INPUT type = text name = nume> <BR>
        Parola:   
        <INPUT type = password name = parola>
        <BR>
        <INPUT  type = submit
                name = submit
          value = "Trimite!">
      </DIV>
    </FORM>
  </BODY>
</HTML>
```

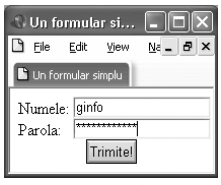

Figura 4

Vom prezenta acum un script care prelucrează datele din formular. Informațiile completate pot fi accesate prin intermediul vectorilor $_POST sau $_REQUEST. Pagina afișată după apăsarea butonului *Trimite!* este prezentată în figura 5.

```
<HTML>
  <HEAD>
    <TITLE>Prelucrare formular</TITLE>
  </HEAD>
  <BODY>
    <?php
      echo "Numele este <B><I> ";
      echo $_POST['nume'];
      echo "</I></B>.<BR>Parola este <B><I> ";
      echo $_REQUEST['parola'];
      echo "</I></B>.<BR>";
    ?>
  </BODY>
</HTML>
```

Figura 5

Există posibilitatea de a afișa atât formularul, cât și rezultatul prelucrării datelor, folosind același *script PHP*. Pentru aceasta, va trebui să verificăm la început "starea curentă a *script*-ului". Dacă formularul a fost completat vom afișa rezultatele prelucrării, în caz contrar vom afișa formularul propriu-zis pentru a permite introducerea datelor.

O posibilitate de determinare a stării curente este includerea unui câmp ascuns în formular, căruia i se atribuie o anumită valoare în momentul în care este completat formularul. Așadar, la începutul *script*-ului vom verifica valoarea acestui câmp ascuns.

Vom crea un singur fișier (form.php), iar atributul **action** al marcajului <**FORM**> va avea tot valoarea form.php:

```
<HTML>
  <?php
    if ($_POST['actiune'] == 'trimis'){
  ?>
  <HEAD>
    <TITLE>Prelucrare formular</TITLE>
  </HEAD>
  <BODY>
    <?php
      echo "Numele este <B><I> ";
      echo $_POST['nume'];
      echo "</I></B>.<BR> Parola este <B><I> ";
      echo $_REQUEST['parola'];
      echo "</I></B>.<BR>";
    ?>
  </BODY>
  <?php
    }
    else{
  ?>
  <HEAD>
    <TITLE>Un formular simplu</TITLE>
  </HEAD>
  <BODY>
    <FORM action = 'form.php' method = post>
      <DIV align = center>
        Numele: 
        <INPUT type = text name = nume><BR>
        Parola:   
        <INPUT type = password name = parola>
        <BR>
        <INPUT type = hidden name = actiune
                                value = trimis>
        <INPUT type = submit name = submit
                              value = "Trimite!">
      </DIV>
    </FORM>
  </BODY>
  <?php
  }
  ?>
</HTML>
```

După prima accesare a documentului **form.php** va fi afișată pagina din figura 4. După apăsarea butonului *Trimite!* va fi accesat același document, dar va fi afișată pagina din figura 5.

În cazul în care butonul prin intermediul căruia sunt trimise informațiile completate este, de fapt, o imagine (inserată printr-o secvență de tipul <**INPUT type** = "image" **src** = "imagine.gif" **name** = "sub">) atunci, pe lângă variabilele corespunzătoare câmpurilor, *script*-ul *PHP* va putea utiliza alte două variabile (sub_x și sub_y) ale căror valori reprezintă coordonatele punctului din imagine, pe care se afla cursorul *mouse*-ului în momentul în care s-a efectuat *click*-ul prin intermediul căruia a fost comunicată programului de navigare intenția de transferare a informațiilor completate.

Constante în PHP

La fel ca în majoritatea limbajelor de programare, constantele sunt identificatori care sunt folosiți pentru accesarea unei valori care nu se poate modifica. Denumirea constantelor respectă aceleași reguli ca și denumirea variabilelor, dar numele lor nu trebuie precedate de semnul '$'.

În *PHP*, constantele pot fi definite folosind funcția define(). Nu este posibilă definirea de constante prin intermediul unor atribuiri simple. Pentru a defini o constantă a cărei valoare este Hello World!, putem folosi secvența: <?**php** define("MESAJ, "Hello World!"); ?>.

Așadar, valoarea constantei MESAJ va fi întotdeauna Hello World!.

Există o mulțime de constante predefinite care pot fi accesate de *script*-urile *PHP*. Acestea se împart în două categorii: constante standard și constante specifice nucleului *PHP* folosit.

Vom prezenta valorile acestor constante în momentul în care le vom folosi.

În *PHP* pot fi definite doar constante care reprezintă valori logice, numere întregi, numere reale sau șiruri de caractere.

Tipuri de date în PHP

Există opt tipuri elementare, grupate în trei categorii. Patru dintre acestea sunt tipuri scalare (`boolean`, `integer`, `float` și `string`), două sunt tipuri compuse (`array` și `object`), iar alte două sunt tipuri speciale (`resource` și `NULL`).

De asemenea, din motive de lizibilitate, au fost introduse trei pseudotipuri: `mixed`, `number` și `callback`. Mai există și tipul `double`, dar semnificația acestuia este aceeași cu cea a tipului `float`. Cele două denumiri coexistă doar din motive "istorice".

În *PHP*, de obicei, tipul unei variabile nu este specificat de către programator, ci este stabilit în timpul execuției în funcție de contextul în care este folosită variabila.

Tipul boolean

Variabilele de acest tip pot avea doar două valori: `ADEVĂRAT` sau `FALS`. Aceste valori pot fi indicate prin cuvintele cheie **`TRUE`** sau **`FALSE`** (pentru ambele nu se face distincție între literele mici și literele mari).

Există posibilitatea de a converti o variabilă de orice tip la tipul `boolean`. În momentul efectuării unei conversii, sunt convertite la valoarea **`FALSE`** următoarele valori:

- numărul întreg `0`;
- numărul real `0.0`;
- șirul vid;
- șirul `"0"`;
- un vector fără nici un element;
- un obiect fără nici o variabilă membru;
- o variabilă de tipul `NULL`;
- o variabilă nedefinită.

Orice altă valoare este convertită la valoarea **`TRUE`** (inclusiv resursele).

Tipul integer

O variabilă de tip `integer` reprezintă o valoare din mulțimea numerelor întregi. Aceste numere pot fi specificate în baza 10, în baza 16 sau în baza 8, convențiile fiind aceleași ca și în limbajele *C/C++* sau *Java*.

Modul de reprezentare depinde de platforma utilizată; de obicei se folosește reprezentarea pe 32 de biți.

Interpretorul *PHP* nu oferă suport pentru numere întregi fără semn.

Trebuie remarcat faptul că în *PHP* nu există nici un operator pentru efectuarea de împărțiri întregi. De exemplu, rezultatul operației `3 / 2` nu va fi un numărul întreg `1` (ca în *C/C++* sau *Java*), ci numărul real (`float`) `1.5`.

Și pentru numerele întregi există posibilitatea efectuării de conversii:

- valoarea logică **`TRUE`** este convertită la valoarea întreagă `1`;
- valoarea logică **`FALSE`** este convertită la valoarea întreagă `0`;
- un număr real este convertit prin "rotunjire înspre 0"; așadar, valoarea reală `2.5` va fi convertită la valoarea întreagă `2`, în timp ce valoarea reală `-2.5` va fi convertită la valoarea întreagă `-2`;
- un șir de caractere este convertit luând în considerare doar primele caractere care conțin informații numerice; așadar șirul `"10"` va fi convertit la valoarea întreagă `10`; de asemenea șirul `"10 ani"` va fi convertit tot la valoarea `10`; dacă primele caractere nu conțin informații numerice, rezultatul conversiei va fi valoarea `0`.

Tipul float

O variabilă de tip `float` poate fi specificată folosind fie forma zecimală, fie cea științifică (cu exponent). La fel ca și în cazul tipului `integer`, precizia variabilelor de tipul `float` este dependentă de platforma utilizată. De obicei se folosește standardul *IEEE* 64.

Există posibilitatea de a converti o variabilă de orice tip la tipul `float`. Pentru numerele reale se pot efectua următoarele conversii:

- un șir de caractere este convertit luând în considerare doar primele caractere care conțin informații numerice; așadar șirul `"10.2"` va fi convertit la valoarea reală `10.2`; șirul `"1.23E1 ani"` va fi convertit la valoarea `12.3`;
- în toate celelalte cazuri se realizează conversii la numere întregi care apoi sunt convertite la valorile reale corespunzătoare.

Tipul string

O variabilă de tip `string` reprezintă un șir de caractere. Un caracter se reprezintă pe un octet, deci sunt 256 de caractere distincte. Acest lucru implică faptul că interpretorul *PHP* nu oferă suport nativ pentru setul de caractere *Unicode*. Lungimea variabilelor de tip `string` nu este limitată de către interpretor.

Literalii de tip șir de caractere pot fi specificați în trei moduri diferite:

- prin folosirea ghilimelelor simple (exemplu `$a = 'acesta este un șir de caractere'`). Pentru a putea avea în cadrul șirului de caractere simbolul `"'"`, atunci înaintea acestuia trebuie scris caracterul `"\"`, iar pentru a putea specifica simbolul `"\"` acesta trebuie dublat. De exemplu, rezultatul instrucțiunii **`echo`** `'acesta este un șir de caractere în care apar caracterele "\\" și "\'"'` este: `acesta este un șir de caractere în care apar caracterele "\" și "'"`;
- prin folosirea ghilimelelor duble. Folosind această notație pot fi specificate mai multe caractere speciale, pe lângă caracterele de la varianta anterioară, printre care: sfârșit de linie (`"\r"`), rând nou (`"\n"`), tab orizontal (`"\t"`), semnul dolar (`"\$"`), ghilimele duble (`"\""`), secvențe de caractere pentru specificarea faptului că o expresie regulară este în notație octală (`"\[0-7]{1,3}"`) și secvențe de caractere pentru specificarea faptului că o expresie

regulară este în notație hexazecimală("\x[0-9A-Fa-f]{1,2}"). Cel mai important lucru este acela că, folosind acest mod de specificare a literalilor de acest tip, numele de variabile care apar în interior vor fi transformate în valoarea lor. De exemplu, dacă $a este o variabilă de tipul integer și are valoarea 2, atunci șirul de caractere "Variabila a are valoarea $a." va fi transformat în șirul "Variabila a are valoarea 2.".

- notația ***heredoc***. Acest tip de notație a fost introdus la versiunea 4 a interpretorului *PHP*. Pentru a specifica un șir de caractere folosind această notație trebuie utilizat operatorul "<<<" urmat de un identificator ales de utilizator. Toate caracterele care se află între operatorul "<<<" urmat de un identificator pe o singură linie, și același identificator pe o altă linie vor constitui valoarea șirului de caractere. De exemplu, instrucțiunea:

```
$str = <<<SF
Acesta este un exemplu
de utilizare a sintaxei
heredoc
SF;
```

va avea ca rezultat un șir de caractere format din trei linii de text.

Pentru a accesa un anumit caracter din șirul de caractere se folosește, după numele variabilei de tip string, indicile caracterului care trebuie accesat scris între acolade. De exemplu, $str{0} returnează primul caracter din șirul de caractere $str.

În cazul în care dorim să concatenăm două șiruri de caractere vom folosi operatorul ".". Folosirea operatorului "+" nu va concatena cele două șiruri.

Există posibilitatea de a converti o variabilă de orice tip la tipul string. Pentru șirurile de caractere se pot efectua următoarele conversii:

- valoarea logică **TRUE** va fi convertită la șirul "1", iar valoarea logică **FALSE** va fi convertită la șirul vid ("");
- un număr întreg va fi convertit la un șir de caractere care reprezintă valoarea numărului în baza 10;
- un număr real va fi convertit la un șir de caractere care reprezintă notația științifică a acestuia;
- obiectele sunt întotdeauna convertite la șirul "Object";
- variabilele de tipul resource sunt convertite la șirul "Resource id #n", unde n reprezintă un număr unic atașat reursei respective de către interpretorul *PHP*;
- valoarea **NULL** este convertită la șirul vid ("").

Tipul array

Vectorii în *PHP* sunt niște mulțimi formate din *chei*. Fiecărei chei din vector i se atașează o valoare. Acest tip de date este optimizat astfel încât să poată fi folosit în locul următoarelor structuri de date: liste, tabele de dispersie, dicționare, colecții, stive, cozi și altele. Datorită faptului că o valoare poate fi reprezentată de un alt vector, se pot simula foarte ușor arborii *n*-dimensionali sau tablourile *n*-dimensionale.

Valoarea unei variabile de tip vector se poate specifica folosind construcția array(cheie => valoare, cheie => valoare, …).

De exemplu, următoarea instrucțiune *PHP* va construi un vector cu două elemente, dintre care unul este de tip string, iar celălat de tip boolean:

```
$a = array("ch" => "string", 12 => TRUE);
```

Variabila $a reprezintă un vector; $a["ch"] are valoarea string, iar $a[12] are valoarea **TRUE**.

În cazul în care nu se specifică o cheie pentru o valoare, atunci acea valoare va fi atașată unei chei care va fi cheia maximă de tip integer folosită anterior, la care se adaugă valoarea 1. Cheile pot avea și valori negative. Dacă nu există chei de tip integer, atunci valoarea va fi atașată cheii 0. De exemplu, următoarele două instrucțiuni sunt echivalente:

```
array(5 => 43, 32, 56, "b" => 12);
array(5 => 43, 6 => 32, 7 => 56, "b" => 12);
```

Dacă se folosește valoarea logică **TRUE** ca și cheie, atunci aceasta va fi convertită la cheia de tip întreg 1, iar valoarea **FALSE** va fi convertită la numărul întreg 0.

Nu se pot folosi pentru chei variabile de tipul array sau object.

O variabilă de tip array se poate modifica prin setarea explicită de valori. De exemplu instrucțiunea $a["x"] = 42; adaugă în vectorul $a valoarea 42 atașată cheii "x".

Dacă se folosește un vector care nu a fost definit anterior, atunci acesta este creat automat. Așadar, printr-o instrucțiune de forma $a[5] = 42, în cazul în care vectorul $a nu există, atunci se va crea un vector cu un singur element. Cheia acestuia va fi numărul întreg 5, iar valoarea sa va fi 42.

De asemenea, există posibilitatea de a crea un element nou fără a-i preciza cheia. Sintaxa are forma $vector[] = valoare; această instrucțiune are ca efect adăugarea unui element a cărui cheie este un număr întreg mai mare cu 1 decât cel mai mare număr întreg care este cheie a unui alt element al vectorului. Dacă nu există nici o astfel de cheie, atunci noul element va avea cheia 0. De exemplu, următoarele două secvențe sunt echivalente:

```
$a[5] = 1;          $a[5] = 1;
$a[6] = 2;          $a[] = 2;
```

Prin conversia la un vector a unei variabile de tip scalar (boolean, integer, float, string) sau resource se creează un vector cu un singur element; cheia acestui element este numărul întreg 0, iar valoarea este cea a variabilei convertite.

Dacă se convertește un obiect (variabilă de tip object), atunci vectorul rezultat va conține câte un element pentru fiecare variabilă membru a obiectului. Cheile elementelor vor fi date de denumirile proprietăților obiectului (variabilele membru ale obiectului), iar valorile elementelor vor fi valorile proprietăților obiectului.

Dacă realizăm o conversie a unei variabile de tip NULL, atunci rezultatul va fi un vector vid (care nu conține nici un element).

Vom prezenta în continuare câteva exemple în care vom descrie mai detaliat posibilitățile oferite de folosirea vectorilor în *PHP*.

Pentru început, vom prezenta un vector ale cărui elemente reprezintă caracteristicile unei portocale. Veți observa că există mari deosebiri dintre vectorii din *PHP* și cei din *Pascal* sau *C/C++*. De fapt, acest vector este echivalent cu tipul înregistrare din aceste limbaje. Așadar, vectorul care conține caracteristicile portocalei poate fi definit astfel:

```
$a = array('denumire' => 'portocala',
           'familie' => 'citrice',
           'culoare' => 'portocaliu',
           'forma' => 'rotunda',
           'gust' => 'dulce'
          );
```

Putem adăuga și alte elemente, care să reprezinte diferite alte proprietăți. De exemplu, am putea avea nevoie de o valoare suplimentară căreia nu dorim să îi atribuim nici un nume de identificare (cheie). Pentru ca vectorul să conțină un element suplimentar cu valoarea `4`, vom putea defini vectorul astfel:

```
$v = array('denumire' => 'portocala',
           'familie' => 'citrice',
           'culoare' => 'portocaliu',
           'forma' => 'rotunda',
           'gust' => 'dulce',
           4
          );
```

Cheia elementului cu valoarea `4` va fi numărul întreg `0` deoarece nu există nici o altă cheie care este număr întreg. O alternativă de construire a acestui vector este următoarea:

```
$v['denumire'] = 'portocala';
$v['familie' = 'citrice';
$v['culoare' = 'portocaliu';
$v['forma' = 'rotunda';
$v['gust' = 'dulce';
$v[] = 4;
```

Putem construi și vectori similari celor din *Pascal* sau *C/C++*.

De exemplu, un vector care conține primele cinci numere naturale strict pozitive și are indicii cuprinși între `0` și `4` poate fi construit astfel.

```
$v[] = 1;
$v[] = 2;
$v[] = 3;
$v[] = 4;
$v[] = 5;
```

Construcțiile echivalente din *Pascal* și *C/C++* sunt următoarele:

Varianta Pascal:

```
const v:array[0..4] of Longint = (1, 2, 3, 4, 5);
```

Varianta C/C++:

```
long v[5] = {1, 2, 3, 4, 5};
```

În *PHP* există și alte modalități de creare a aceluiași vector. Două dintre acestea sunt: `$v = array(0 => 1, 1 => 2, 2 => 3, 3 => 4, 4 => 5);` și `$v = array(1, 2, 3, 4, 5);`.

Evident, există posibilitatea de a crea vectori care au ca elemente valori care nu respectă nici o regulă aparentă. Iată un exemplu:

```
$v = array(10,          // cheia este 0
           5 => 6,      // cheia este 5
           3 => 7,      // cheia este 3
           'a' => 4,    // cheia este 'a'
           11,          // cheia este 6
           '8'  =>  2,  // cheia este 8 (întreg!)
           '02' => 77,  // cheia este '02'
           0 => 12      // cheia este 0; valoarea 12
                        //   suprascrie valoarea 10
          );
```

Putem crea și vectori vizi (care nu conțin nici un element); pentru aceasta vom folosi o instrucțiune de tipul: `$vector_vid = array();`.

Elementele unui vector pot fi considerate ca reprezentând o colecție de date.

Următorul exemplu ilustrează modul de utilizare a unui astfel de vector (pagina *Web* generată este prezentată în figura 6):

```
<HTML>
  <HEAD>
    <TITLE>Colection</TITLE>
  </HEAD>
  <BODY>
    <?php
      $colors = array('red',
                      'green',
                      'blue',
                      'silver'
                     );
      foreach ($colors as $color){
        echo "<FONT size = 4 color = $color>";
        echo "<B>This text is <I>$color!</I>";
        echo "</B></FONT><BR>";
      }
    ?>
  </BODY>
</HTML>
```

Figura 6

Nu este posibilă modificarea valorilor unui vector folosind un astfel de ciclu. O posibilitate de a rezolva această problemă este folosirea unei secvențe de tipul celei prezentate în exemplul următor.

```
<HTML>
  <HEAD>
    <TITLE>Modificare</TITLE>
  </HEAD>
  <BODY>
    <?PHP
      $colors = array('red', 'green', 'blue',
                                      'silver');
      foreach ($colors as $key => $color){
        $colors[$key] = strtoupper($color);
      }
      echo "<TT>";
      print_r($colors);
      echo "</TT>"
    ?>
  </BODY>
</HTML>
```

Pagina *Web* generată este asemănătoare cu cea prezentată în figura 7.

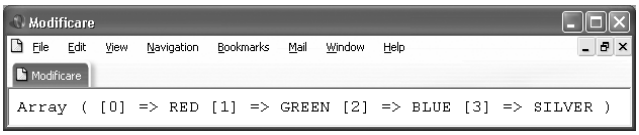

Figura 7

Vom prezenta acum un exemplu în care un vector este completat cu numele fișierelor și subdirectoarelor din directorul curent. Aceste denumiri vor fi afișate într-o pagină *Web* (prezentată în figura 8).

```
<HTML>
  <HEAD>
    <TITLE>Fisiere</TITLE>
  </HEAD>
  <BODY>
    <?PHP
      $handle = opendir('.');
      while ($file =
              readdir($handle)){
        $files[] = $file;
      }
      echo "<TT>";
      foreach ($files as $file){
        echo "$file<BR>";
      }
      closedir($handle);
      echo "</TT>";
    ?>
  </BODY>
</HTML>
```

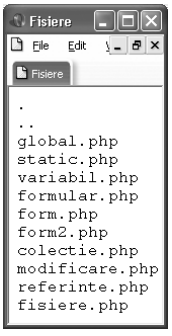

Figura 8

Datorită faptului că valoarea unui element al unui vector poate avea orice tip, elementele pot fi vectori. Așadar, pot fi create tablouri multidimensionale care au orice structură dorită. Un exemplu este prezentat în continuare:

```
fructe = array("fructe" => array("a" => "mere",
                                 "b" => "pere",
                                 "c" => "alune"
                                 ),
               "numere" => array(1, 2, 3, 4, 5, 6),
               "metale" => array("platină",
                                  5 => "aur",
                                  "argint"
                                 )
              );
```

În aceste condiții, de exemplu, comanda **echo** `$fructe["metale"][5];` duce la afișarea șirului `aur`. De asemenea, comanda **echo** `$fructe["fructe"]["b"];` determină afișarea șirului `pere`.

Pot fi create direct și tablouri multidimensionale prin instrucțiuni de tipul `$suc["mere"]["verde"]="gustos";`.

Tipul object

Vom prezenta într-un articol viitor mai multe detalii referitoare la obiectele *PHP*. Acum vom face doar câteva precizări generale referitoare la acest tip de date.

Pentru a defini un obiect care poate fi folosit pentru afișarea mesajului `Hello World!` scrie următoarea secvență:

```
class HelloWorld{
  function DisplayHelloWorld(){
    echo "Hello World!";
  }
}
```

Pentru a utiliza o variabilă de tip obiect va trebui să realizăm o ***instanțiere*** prin intermediul instrucțiunii **new**; sintaxa este următoarea:

```
$a = new HelloWorld;
```

Astfel, variabila `$a` devine un obiect ale cărui metode pot fi utilizate. Pentru afișarea propriu-zisă a mesajului va trebui să executăm metoda `DisplayHelloWorld()` printr-o instrucțiune de tipul `$a->DisplayHelloWorld();`.

Orice variabilă de un anumit tip poate fi convertită într-un obiect. Dacă variabila respectivă este un obiect, atunci ea nu va fi modificată. În caz contrar, efectul conversiei este crearea unei noi instanțe a clasei `stdClass`. Dacă variabila are tipul `NULL`, atunci noua instanță va fi vidă. În toate celelalte cazuri instanța va conține o variabilă membru numită scalar a cărei valoare va fi cea a variabilei convertite. Pentru conversii vom folosi instrucțiuni de tipul `$obiect = (object)"Hello World!"`. După realizarea conversiei vom putea tipări mesajul `Hello World!` folosind instrucțiunea **echo** `$obiect->scalar;`.

Tipul resource

Variabilele de tip `resource` sunt folosite pentru păstrarea unor referințe către anumite resurse externe cum ar fi conexiuni la baze de date, fișiere etc. Resursele sunt create și utilizate de anumite funcții speciale. Datorită specificului acestui tip de date valoarea nici unei variabile de alt tip nu poate fi convertită la tipul `resource`.

Tipul NULL

Valoarea specială `NULL` este atribuită oricărei variabile care nu a fost inițializată. Această valoare este singura pe care o pot avea variabilele de tip `NULL`. Se consideră că o variabilă are tipul `NULL` dacă:

- i s-a atribuit constanta `NULL`;
- nu a fost inițializată;
- a fost dezinițializată (prin intermediul funcției `unset()`).

Tipul unei variabile

Interpretorul *PHP* nu necesită (de fapt nici nu permite) definirea explicită a tipului unei variabile. Acest tip este determinat în funcție de contextul în care este utilizată variabila. Așadar, dacă unei anumite variabile i se atribuie un șir de caractere, tipul ei devine `string`; dacă, ulterior, i se atribuie un număr întreg, tipul ei devine `integer`.

Interpretorul *PHP* este capabil să realizeze conversii automate în unele situații. Un exemplu în acest sens este utilizarea operatorului de adunare (`+`). Dacă unul dintre operanzi are tipul `float`, atunci toți operanzii sunt convertiți în numere reale, iar rezultatul este un număr real. În toate celelalte cazuri operanzii sunt convertiți în numere întregi, rezultatul fiind un număr întreg. Trebuie menționat faptul că tipurile operanzilor ***nu se schimbă***, singura modificare este legată de modul în care sunt interpretate valorile acestora în momentul efectuării adunării.

Dacă dorim ca o anumită variabilă să fie convertită la un anumit tip, atunci putem utiliza operatorii de conversie care vor fi descriși în cadrul acestui articol. Este posibil să modificăm explicit tipul unei variabile prin intermediul funcției `settype()`. Funcția pereche (`gettype()`) returnează tipul pe care îl are o anumită variabilă în momentul apelării acestei funcții.

Realizarea conversiilor

Așa cum am afirmat anterior, pentru a impune considerarea unei variabile ca având un anumit tip vom putea folosi operatorii de conversie. Modul de utilizare este foarte simplu: pentru a realiza conversia unei variabile la un anumit tip, aceasta va fi precedată de operatorul de conversie; între operator și variabilă pot apărea spații. Lista operatorilor de conversie este prezentată în continuare:

- (**`int`**) sau (**`integer`**) - variabila este considerată a fi un număr întreg;
- (**`bool`**) sau (**`boolean`**) - variabila este considerată a fi o valoare logică (**`TRUE`** sau **`FALSE`**);
- (**`float`**), (**`double`**) sau (**`real`**) - variabila este considerată a fi un număr real;
- (**`string`**) - variabila este considerată a fi un șir de caractere;
- (**`array`**) - variabila este considerată a fi un vector;
- (**`object`**) - variabila este considerată a fi un obiect;

Imediat după paranteza deschisă și imediat înaintea parantezei închise este permisă apariția spațiilor. Așadar construcțiile `$a = (int) $b` și `$a = ( int ) $b` sunt echivalente.

Expresii în PHP

Cea mai intuitivă definiție a unei expresii este dată de sintagma "*orice are o valoare*". Cele mai simple expresii sunt cele formate din variabile sau constante. De exemplu, constanta `5` este o expresie a cărei valoare este `5`, o variabilă căreia i s-a atribuit valoarea `6` este o expresie a cărei valoare este `6` etc.

O altă categorie de expresii este reprezentată de funcții. Acestea returnează întotdeauna o valoare care este considerată a fi valoarea expresiei corespunzătoare apelului funcției.

Cu ajutorul operatorilor pot fi construite expresii mai complexe. Majoritatea operatorilor sunt binari (noua expresie este formată din două subexpresii și un operator). Există mai mulți operatori unari (expresia este formată din operator și o altă expresie) și un operator ternar (expresia este formată din operator și trei subexpresii).

Marea majoritate a operatorilor sunt preluați din limbajul C++; în cadrul articolului din numărul următor vom prezenta detaliat operatorii pe care îi avem la dispoziție în *PHP*.

Concluzii

În cadrul acestui articol am prezentat câteva elemente de bază ale limbajului *PHP*.

Am descris modul în care este inserat codul *PHP* în documentele care duc la generarea de pagini *Web*, precum și sintaxa generală a unei instrucțiuni *PHP*.

În continuare am prezentat detalii referitoare la utilizarea variabilelor, constantelor, tipurilor de date și a expresiilor. De asemenea, au fost prezentate modalitățile prin care o variabilă de un anumit tip poate fi convertită la un alt tip.

În cadrul articolului am folosit unele elemente simple (de exemplu, anumite funcții) care nu au fost prezentate anterior. Acestea au fost utilizate pentru ca exemplele să fie mai interesante și nu împiedică înțelegerea mecanismelor care stau la baza utilizării elementelor prezentate. În numerele următoare vom prezenta detaliat toate aceste elemente.

În următorul articol vom descrie operatorii care pot fi folosiți în *PHP*, precum și principalele structuri de control puse la dispoziție de acest interpretor.

Mihai Scorțaru este redactor-șef GInfo și poate fi contactat prin e-mail la adresa skortzy@yahoo.com. *Claudiu Soroiu este redactor GInfo și poate fi contactat prin e-mail la adresa* csoroiu@yahoo.com.