Instrucțiuni *PHP*

Structurile de control sunt acele structuri din cadrul unui limbaj care permit controlul fluxului de execuție într-un program sau *script*. Aceste structuri pot fi:

- condiționale
- repetitive

Instrucțiuni condiționale

Instrucțiunea *if – elseif - else*

Condițiile exprimate în instrucțiunea *if* trebuie să fie cuprinse între paranteze ().

De exemplu, dacă vizitatorul site-ului nu comandă niciun produs, probabil că butonul *Submit* a fost apăsat din greşeală; prin urmare pagina ar trebui să afişeze un mesaj corespunzător în loc de „Order processed”:

```
if( $totalqty == 0 )
   echo 'You did not order anything on the previous page!<br />';
```

Blocuri de cod

Pot exista mai multe instrucțiuni care trebuie executate în funcție de condițiile unei instrucțiuni condiționale. În acest caz, instrucțiunile vor fi grupate în blocuri, cuprinse între acolade {}.

```
if ($totalqty == 0) {
     echo '<p style="color:red">';
     echo 'You did not order anything on the previous page!';
     echo '</p>';
}
```

Ramura *else* a instrucțiunii *if* permite definirea unei alternative care se execută atunci când condiția exprimată în instrucțiunea *if* este *false*. De

exemplu, afişăm un mesaj dacă un client nu comandă nimic, iar în caz contrar afişăm numărul de produse comandate din fiecare categorie:

```
if ($totalqty == 0) {
   echo "You did not order anything on the previous page!<br />";
} else {
   echo $tireqty." tires<br />";
   echo $oilqty." bottles of oil<br />";
   echo $sparkqty." spark plugs<br />";
}
```

În cazul multor decizii există mai mult de 2 opțiuni. Se poate creea o secvență corespunzătoare acestor opțiuni utilizând clauza *elseif*. În acest caz, programul verifică fiecare condiție până când găseşte una care este adevărată.

Exemplu: În magazin se aplică reduceri pentru comenzile mari de cauciucuri, astfel:

- mai puțin de 10 cauciucuri – nicio reducere
- 10-49 cauciucuri – 5% reducere
- 50-99 cauciucuri – 10% reducere
- >= 100 cauciucuri – 15% reducere

Secvența de cod care stabileşte valoarea reducerii este următoarea:

```
if ($tireqty < 10) {
      $discount = 0;
} elseif (($tireqty >= 10) && ($tireqty <= 49)) {
      $discount = 5;
} elseif (($tireqty >= 50) && ($tireqty <= 99)) {
      $discount = 10;
} elseif ($tireqty >= 100) {
      $discount = 15;
}
```

Observație: Formele *elseif* şi *else if* sunt ambele corecte.

Instrucțiunea *switch*

Această instrucțiune lucrează într-un mod similar instrucțiunii *if*, dar permite condiției să ia mai mult de 2 valori. Într-o instrucțiune *switch*, condiția poate lua orice număr de valori diferite, atât timp cât este evaluată la un tip simplu (*integer*, *string* sau *float*). Trebuie furnizată o clauză *case* care să trateze fiecare valoare căreia îi va corespunde o acțiune şi, opțional, o clauză *case* implicită care să trateze cazurile neprevăzute explicit.

De exemplu, dorim să cunoaştem formele de publicitate care au adus clienți, deci putem adăuga o întrebare în formular, astfel încât acesta va arăta astfel: http://193.226.51.37/web/curs2/exemplu1.html

```
<tr>
<td>How did you find us?</td>
<td><select name="find">
<option value = "a">I'm a regular customer</option>
<option value = "b">TV advertising</option>
<option value = "c">Phone directory</option>
<option value = "d">Word of mouth</option>
</select>
</td>
</tr>
```

Codul HTML precedent adaugă o nouă variabilă formularului (*find*) ale cărei valori vor fi *a*, *b*, *c* sau *d*. Variabila poate fi prelucrată cu ajutorul mai multor clauze *elseif* sau printr-o instrucțiune *switch*:

```
switch($find) {
case "a" :
echo "<p>Regular customer.</p>";
break;
case "b" :
echo "<p>Customer referred by TV advert.</p>";
break;
case "c" :
echo "<p>Customer referred by phone directory.</p>";
break;
case "d" :
echo "<p>Customer referred by word of mouth.</p>";
break;
default :
echo "<p>We do not know how this customer found us.</p>";
break;
}
```

Observații:

- Exemplul anterior presupune că variabila *$find* a fost extrasă din tabloul *$_POST*.
- Comportamentul instrucțiunii *switch* este diferit de cel al lui *if-elseif*: instrucțiunea *if* afectează o singură clauză, în vreme ce selectarea unei clauze *case* a instrucțiunii *switch* conduce la executarea de către PHP a instrucțiunilor până când este întâlnită instrucțiunea *break*, care determină trecerea la următoarea instrucțiune după *switch*. Dacă

instrucțiunea *break* lipseşte, *switch* va executa tot codul care urmează clauzei *case* a cărei condiție a fost adevărată.

Instrucțiuni iterative

Exemplu: Dorim afişarea unui tabel care să prezinte costul de transport corespunzător comenzilor. Costul depinde de distanța la care are loc livrarea şi poate fi deteminat pe baza unei formule.

Tabelul este următorul: http://193.226.51.37/web/curs2/transport.html

Codul HTML care afişează acest tabel este următorul: http://193.226.51.37/web/curs2/transport.txt

Se observă că acest cod este lung şi repetitiv. În locul acestuia, putem folosi instrucțiuni PHP de ciclare.

Instrucțiunea *while*

Forma sintactică a acestei comenzi este următoarea:

```
while( condition ) expression;
```

Exemplu: Să se afişeze primele 5 numere întregi pozitive.

```
$num = 1;
while ($num <= 5 ){
      echo $num."<br />";
$num++;
}
```

Exemplu: Afişarea costurilor de transport corespunzătoare comenzilor se poate realiza cu ajutorul instrucțiunii *while* (http://193.226.51.37/web/curs2/transport.php), codul sursă corespunzător fiind http://193.226.51.37/web/curs2/transport_php.txt

Instrucțiunile *for* şi *foreach*

Forma sintactică a comenzii de ciclare *for* este:

```
for( expression1; condition; expression2)
      expression3;
```

- *expression1* este evaluată o singură dată, la început, stabilind valoarea inițială a contorului;
- expresia *condition* este testată înainte de fiecare iterație; dacă returnează *false*, iterația se încheie;

- *expression2* este executată o dată la sfârşitul fiecărei iteraţii, actualizând valoarea contorului;
- *expression3* se execută pentru fiecare iteraţie, fiind reprezentată de un bloc de cod.

Exemplu: Calculul costurilor de transport cu ajutorul instrucţiunii *for* este http://193.226.51.37/web/curs2/transport_for.txt

Observaţie: Variabilele variabile pot fi utilizate împreună cu instrucţiunea *for* pentru a itera printr-o mulţime de câmpuri ale unui formular. Dacă numele câmpurilor sunt *name1*, *name2*, ... acestea pot fi gestionate astfel:

```
for ($i=1; $i <= $numnames; $i++){
   $temp= “name$i”;
   echo $$temp.’<br />’; // sau o alta procesare
}
```

Pe lângă instrucţiunea *loop*, mai există o instrucţiune de iterare specifică lucrului cu tablouri, *foreach*.

Instrucţiunea *do...while*

Forma sintactică a acestei instrucţiuni este următoarea:

```
do
     expression;
while( condition );
```

Spre deosebire de instrucţiunea *while*, condiţia este testată la sfârşit, astfel că instrucţiunea sau blocul de cod din *do...while* se execută cel puţin o dată.

Instrucţiuni de ieşire

Dacă dorim oprirea execuţiei une secvenţe de cod, avem la dispoziţie 3 abordări.

- Instrucţiunea *break* opreşte execuţia unei comenzi de ciclare.
- Instrucţiunea *continue* conduce la trecerea la următoarea iteraţie.
- Instrucţiunea *exit* conduce la terminarea întregului script PHP. Acest lucru este util atunci când se realizează verificarea erorilor.

Utilizarea structurilor alterative de control

Pentru toate structurile de control prezentate, există o formă sintactică alternativă. Aceasta constă în înlocuirea acoladei deschise ({) cu „:” iar a celei închise (}) cu un cuvânt cheie care va fi *endif*, *endswitch*, *endwhile*,

endfor sau *endforeach*, în funcție de tipul instrucțiunii. Nu există o sintaxă alternativă pentru instrucțiunea *do...while*.

Exemplu: Codul sursă:

```
if ($totalqty == 0) {
   echo "You did not order anything on the previous page!<br />";
   exit;
}
```

poate fi scris sub forma următoare:

```
if ($totalqty == 0) :
   echo "You did not order anything on the previous page!<br />";
   exit;
endif;
```

Utilizarea cuvântului cheie declare

Forma sintactică a acestei structuri de control din PHP este următoarea:

```
declare (directive)
{
   // block
}
```

Această structură este folosită pentru stabilirea directivelor de execuție pentru un bloc de cod, adică a unor reguli asupra modului în care codul urmează să fie executat. Până acum a fost implementată o singură directivă de execuție, denumită *ticks*. Aceasta se setează *ticks* = *n* şi permite ca o anumită funcție să fie executată la fiecare *n* linii de cod din interiorul unui bloc, lucru care este util în activitatea de *debug*.

Stocarea şi regăsirea datelor

Datele introduse prin intermediul unui formular trebuie stocate pentru prelucrarea lor ulterioară. Vom învăța cum să salvăm aceste date într-un fişier iar apoi într-o bază de date *MySQL*. În privința fişierelor, vom studia operațiile care pot fi realizate asupra acestora.

Salvarea datelor

În exemplul din această secțiune vom scrie comenzile clienților într-un fişier text, câte o comandă pe fiecare linie. Scrierea comenzilor în acest mod este foarte simplă, însă are nişte limite considerabile. Atunci când este gestionat un volum considerabil de informații, va trebui utilizată o bază de date.

Scrierea şi citirea în/din fişiere este foarte similară multor limbaje de programare.

Stocarea şi regăsirea comenzilor

Vom utiliza următoarea versiune (uşor modificată) a formularului pe care am lucrat până acum: http://193.226.51.37/web/curs2/orderform2.html

Formularul a fost modificat astfel încât include un mod rapid de regăsire a adresei de livrare. Noul câmp se numeşte *address* şi îi corespunde o variabilă care poate fi accesată ca *$_REQUEST['address']*, *$_POST['address']* sau *$_GET'['address']*, în funcție de metoda aleasă.

Vom scrie toate comenzile lansate de către clienți într-un fişier, apoi vom construi o interfață web pentru vizualizarea acestor comenzi.

Procesarea fişierelor

Scrierea datelor într-un fişier presupune 3 paşi:

1. Deschiderea fişierului; dacă acesta nu există, va trebui creat.
2. Scrierea datelor în fişier.
3. Închiderea fişierului.

Citirea dintr-un fişier presupune:

1. Deschiderea fişierului; trebuie prevăzută eroarea care apare în cazul în care deschiderea nu se poate realiza (de exemplu, fişierul nu există).
2. Citirea datelor din fişier.
3. Închiderea fişierului.

Deschiderea fişierelor

Se realizează cu ajutorul funcției *fopen*(). La deschiderea unui fişier, trebuie specificat şi modul în care acesta va fi utilizat. Modul furnizează sistemului de operare un mecanism prin care determină cum să trateze accesul din partea altor persoane sau script-uri, precum şi o metodă de a verifica dacă avem acces şi drepturi asupra unui anumit fişier.

La deschiderea unui fişier trebuie făcute 3 alegeri:

1. Fişierul poate fi deschis doar pentru citire, doar pentru scriere, sau pentru ambele operații.
2. Dacă scriem într-un fişier, putem opta pentru suprascrierea conținutului sau adăugarea noilor date la sfârşitul fişierului. O altă posibilitate este aceea de a încheia programul dacă fişierul există deja, în loc de a-l suprascrie.
3. Dacă scriem într-un fişier dintr-un sistem care face diferență între fişierele text şi cele binare, va trebui specificat acest lucru.

Funcția *fopen*() suportă combinații ale acestor 3 opțiuni. Deschiderea unui fişier pentru scrierea comenzilor din exemplul nostru se realizează prin:

```
$fp = fopen("$DOCUMENT_ROOT/../orders/orders.txt", 'w');
```

Primul parametru din apelul anterior este calea către fişier; a fost utilizată variabila PHP predefinită $_SERVER['DOCUMENT_ROOT']. Calea furnizată este una relativă: utilizăm „..” pentru crearea fişierului în directorul părinte al celui specificat în $DOCUMENT_ROOT, deoarece nu dorim ca acest fişier să fie accesibil pe web.

Pentru ca linia anterioară să fie corectă, este necesar să scriem la începutul script-ului:

```
$DOCUMENT_ROOT = $_SERVER['DOCUMENT_ROOT'];
```

Pot fi utilizate căi absolute sau putem să nu specificăm nicio cale, caz în care fişierul va fi creat sau căutat în directorul în care se află script-ul.

Într-un mediu Unix se utilizează caracterul slash (/) în specificarea căilor către directoare. În Windows, se poate utiliza atât slash (/) cât şi backslash (\). În cazul caracterului backslash, acesta trebuie marcat drept caracter special, prin adăugarea încă a unui astfel de caracter (\):

```
$fp = fopen("$DOCUMENT_ROOT\\..\\orders\\orders.txt", 'w');
```

Al doilea parametru al funcției *fopen*() reprezintă modul de deschidere a fişierului. Acesta este un *string* şi poate lua una dintre valorile

- *r* (*read*) – deschiderea fişierului pentru citire, de la începutul fişierului;
- *r+* – deschiderea fişierului pentru citire şi scriere, de la începutul fişierului;
- *w* (*write*) – deschiderea fişierului pentru scriere, de la începutul acestuia; dacă fişierul există, va fi şters conținutul său;
- *w+* – deschiderea fişierului pentru scriere şi citire, de la începutul acestuia; dacă fişierul există, va fi şters conținutul său;
- *x* (*cautious write*) – deschiderea fişierului pentru scriere, de la începutul acestuia; dacă fişierul există, nu va fi deschis, ci *fopen*() returnează *false*, iar PHP generează un *warning*;
- *x+* - deschiderea fişierului pentru scriere şi citire, de la începutul acestuia; dacă fişierul există, nu va fi deschis, ci *fopen*() returnează *false*, iar PHP generează un *warning*;
- *a* (*append*) – deschiderea fişierului doar pentru adăugare, începând de la sfârşitul conținutului curent; dacă fişierul nu există, va fi creat;
- *a+* – deschiderea fişierului pentru adăugare şi citire, începând de la sfârşitul conținutului curent; dacă fişierul nu există, va fi creat;
- *b* (*binary*) – se utilizează în conjuncție cu unul dintre celelalte moduri; este util pentru sistemele care fac diferența între fişierele text şi cele binare (Windows face această diferență, Unix nu). Se recomandă utilizarea opțiunii pentru asigurarea portabilității. Aceasta este valoarea implicită.
- *t* (*text*) – opțiune existentă doar în Windows.

Al treilea parametru al funcției *fopen* este opțional şi se utilizează dacă dorim să căutăm în *include_path* (parametru de configurare al lui PHP). În caz afirmativ, valoarea acestui parametru va fi 1 şi nu va mai fi necesar să specificăm un director sau o cale pentru fişierul respectiv.

```
$fp = fopen('orders.txt', 'ab', true);
```

Al patrulea parametru este, de asemenea, opțional. Funcția *fopen*() permite ca numele fişierelor să fie prefixate de un protocol (de exemplu, http://) şi deschise la o locație distantă. Unele protocoale necesită un parametru. Dacă funcția *fopen*() deschide fişierul cu succes, este returnată o resursă (pointer către fişier) care trebuie stocată într-o variabilă ($fp).

Deschiderea fişierelor prin *FTP* sau *HTTP*

Pe lângă deschiderea fişierelor locale pentru citire sau scriere, pot fi deschise fişiere prin intermediul FTP, HTTP sau al altor protocoale. Această posibilitate poate fi eliminată prin dezactivarea directivei *allow_url_fopen* din fişierul php.ini.

Scrierea într-un fişier

Pentru scrierea într-un fişier în PHP se pot itiliza funcțiile *fwrite*() sau *fputs*(). Aceasta din urmă este un alias al lui *fwrite*(). Un apel la funcția *fwrite* este de forma următoare:

```
fwrite($fp, $outputstring);
```

şi determină scrierea şirului de caractere stocat în *$outputstring* în fişierul la care pointează $*fp*.

O alternativă a lui *fwrite*() este funcția *file_put_contents()*. Aceasta are următorul prototip:

```
int file_put_contents ( string filename,
                        string data
                        [, int flags
                        [, resource context]])
```

Această funcție scrie şirul conținut în parametrul *data* în fişierul numit *filename* fără a fi nevoie de invocarea funcțiilor *fopen*() şi *fclose*(). Parametrii opționali *flags* şi *context* sunt folosiți la scrierea la distanță a fişierelor.

Parametrii funcției *fwrite*

Prototipul funcției *fwrite* este următorul:

```
int fwrite ( resource handle, string string [, int length])
```

Al treilea parametru reprezintă numărul maxim de octeţi care urmează să fie scrişi. Specificarea acestuia determină scrierea lui *string* în fişier până când sunt scrişi *length* octeţi sau până când este scris tot şirul (dacă acesta este mai scurt decât *length*).

Lungimea unui şir de caractere poate fi obţinută cu ajutorul funcţiei *strlen*:

```
fwrite($fp, $outputstring, strlen($outputstring));
```

Formatul fişierelor

Fie următorul şir de caractere, care reprezintă o înregistrare în fişier:

```
$outputstring = $date. "\t".$tireqty. " tires \t".$oilqty. " oil\t"
.$sparkqty. " spark plugs\t\$".$totalamount
. "\t". $address. "\n";
```

Fiecare înregistrare va fi scrisă pe o linie nouă (\n) iar câmpurile sunt separate prin tab (\t).

Închiderea unui fişier

După terminarea lucrului cu un fişier este nevoie să îl închidem:

```
fclose($fp);
```

Această funcţie returnează *true* dacă închiderea fişierului a avut loc cu succes sau *false* altfel.

Forma finală a fişierului *processorder.php* este dată în http://193.226.51.37/web/curs2/processorder2.txt