Pagini WEB cu PHP 4

Mihai Scorțaru, Claudiu Soroiu

În cadrul acestui articol dedicat creării paginilor Web, folosind interpretorul PHP, vom prezenta câteva facilități care pot fi utilizate pentru construirea unor site-uri cu conținut dinamic.

Autentificarea

După cum probabil știți, anumite *site*-uri au accesul restricționat. Așadar, conținutul lor poate fi accesat numai de către anumite persoane autorizate.

Una dintre variantele prin care poate fi verificat faptul dacă un anumit utilizator are dreptul de a accesa un anumit *site* este cererea introducerii unui nume de utilizator și a unei parole în momentul în care persoana respectivă încearcă să acceseze aceste informații.

În momentul accesării paginii *Web* este trimis un mesaj care indică faptul că autentificarea este necesară. Programul de navigare va afișa o casetă de dialog care permite introducerea unui nume de utilizator și a unei parole. După introducerea informațiilor necesare va fi accesată din nou pagina *Web* dar, de această dată, variabilele predefinite `PHP_AUTH_USER`, `PHP_AUTH_PW` și `PHP_AUTH_TYPE` vor conține numele de utilizator, parola, respectiv tipul autentificării.

Pentru a trimite mesajul care cere autentificarea este utilizată funcția `header()`. Nu este posibilă decât autentificarea de tip "***Basic***". Un document *PHP* care impune o autentificare înaintea accesării informațiilor este prezentat în continuare:

```
<?PHP
  if (!isset($_SERVER['PHP_AUTH_USER'])){
    header('WWW-Authenticate: Basic
                              realm="GInfo"');
    header('HTTP/1.0 401 Unauthorized');
    echo 'Pentru accesarea conținutului este
                  necesară o autentificare!';
    exit;
  }
  else{
    echo "<HTML><HEAD>";
    echo "<TITLE>Autentificare</TITLE>";
    echo "</HEAD><BODY>";
    echo "<P>Salut <B><I>
       {$_SERVER['PHP_AUTH_USER']}</I></B>!";
    echo "<P>Parola introdusă este <B><I>
         {$_SERVER['PHP_AUTH_PW']}</I></B>.";
    echo "</BODY></HTML>";
  }
?>
```

Pentru o compatibilitate cât mai mare, cuvântul `Basic` trebuie scris cu majusculă, valoarea pentru `realm` trebuie să fie cuprinsă între ghilimele (nu între apostrofuri), iar între `HTTP/1.0` și `401` trebuie să existe exact un spațiu.

În acest exemplu, numele de utilizator și parola sunt afișate. În practică, se va verifica validitatea lor, de obicei printr-o interogare a unei baze de date.

Figura 1: Opera 7.02

Caseta de dialog care apare în momentul în care este necesară o autentificare diferă de la un program de navigare la altul.

Figura 2: Netscape Navigator 7.02

În figurile 1, 2 și 3 sunt prezentate casetele de dialog afișate de ultimele versiuni ale celor mai cunoscute *browser*-e: *Opera* 7.02, *Netscape Navigator* 7.02 și *Microsoft Internet Explorer* 6.0 *Service Pack* 1.

Figura 3: Internet Explorer 6.0 SP 1

În cazul în care este instrodus un nume de utilizator

și o parolă și este apăsat butonul ***Ok***, aceste informații vor fi afișate. Pagina *Web* generată este prezentată în figura 4.

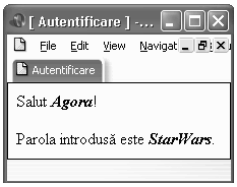

Figura 4

În situația în care este apăsat butonul ***Cancel*** (sau cel echivalent; de exemplu, ***Revocare*** în *Internet Explorer* 6 localizat pentru limba română) pagina *Web* generată va conține un mesaj de avertizare; situația este ilustrată în figura 5.

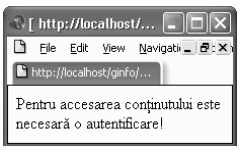

Figura 5

Cookie-uri

În cadrul unui articol dedicat limbajului *JavaScript* am prezentat modul în care pot fi stocate informații pe calculatorul care accesează o anumită pagină *Web* și cum pot fi acestea accesate ulterior.

În *PHP* există o funcție specială care poate fi folosită pentru crearea *cookie*-urilor și anume `setcookie()`. Funcția are șase parametri, doar prezența primului fiind obligatorie:

- `nume`: șir de caractere care reprezintă numele *cookie*-ului;
- `valoare`: șir de caractere care reprezintă valoarea *cookie*-ului (informația corespunzătoare);
- `expirare`: un număr întreg care indică momentul de timp în care *cookie*-ul nu va mai putea fi utilizat;
- `cale`: șir de caractere care reprezintă o cale generică a fișierelor care pot accesa *cookie*-ul;
- `domeniu`: șir de caractere care reprezintă domeniul pe care trebuie stocate fișierele care pot accesa *cookie*-ul;
- `securitate`: număr întreg care, dacă are valoarea `1`, arată că informația poate fi transmisă doar folosindu-se o conexiune *HTTPS* securizată.

Dacă unul dintre parametri este prezent, atunci toți parametrii anteriori trebuie să fie prezenți. Dacă unul dintre aceștia nu este utilizat, se va folosi în poziția respectivă fie șirul vid, fie valoarea `0`.

În continuare vom prezeta câteva exemple de apeluri prin care sunt create *cookie*-uri:

```
setcookie("TestCookie","GInfo");
setcookie("TestCookie","GInfo",time()+3600);
          // expiră după o oră de la creare
setcookie("TestCookie","GInfo",time()+3600,
                "/revista/",".ginfo.ro",1);
```

Cu ajutorul funcției `setcookie()` este posibilă și ștergerea *cookie*-urilor. Următoarele exemple ilustrează modul în care poate fi realizată această operație:

```
setcookie("TestCookie","",time()-3600);
setcookie("TestCookie","",time()-3600,
              "/concurs/", ".ginfo.ro", 1);
```

Se observă că momentul expirării este setat ca fiind anterior momentului executării operației de ștergere.

Există posibilitatea de a crea șiruri de *cookie*-uri prin simpla includere a parantezelor drepte în denumiri. Un exemplu de creare și utilizare a unor astfel de șiruri este prezentat în continuare:

```
setcookie("cookie[three]","cookiethree");
setcookie("cookie[two]","cookietwo");
setcookie("cookie[one]","cookieone");
if (isset($cookie)) {
  while (list($name,$value) = each($cookie)){
    echo "$name == $value<BR>";
  }
}
```

Pentru a accesa *cookie*-urile stocate pe calculatorul unui vizitator al paginii *Web*, este disponibil vectorul `$_COOKIE`. Vom prezenta în continuare un document *PHP* cu ajutorul căruia se generează o pagină care conține denumirile și valorile tuturor *cookie*-urilor.

```
<HTML>
  <HEAD>
    <TITLE>
      Cookie-uri
    </TITLE>
  </HEAD>
  <BODY>
    <?PHP
      foreach ($_COOKIE as $cheie => $valoare)
        echo "$cheie = $valoare<BR>";
    ?>
  </BODY>
</HTML>
```

Încărcarea fișierelor

Fișierele pot fi încărcate pe un *server* dacă utilizatorul folosește un program de navigare compatibil *RFC*-1867; pot fi utilizate ultimele versiuni ale celor mai cunoscute *browser*-e.

Este posibilă încărcarea atât a fișierelor text, cât și a celor binare. Folosind autentificarea *PHP* și funcțiile de manipulare a fișierelor, avem control total asupra utilizatorilor care pot încărca fișiere și asupra operațiilor efectuate cu fișierul încărcat.

Formularul de încărcare

Pentru ca utilizatorul să poată încărca un fișier, el trebuie să aibă la dispoziție un formular care să permită alegerea fișierului.

Un astfel de formular poate fi creat, de exemplu, folosind limbajul *HTML* standard. Prezentăm în continuare un document *HTML* care generează o pagină *Web* care permite alegerea unui fișier și încărcarea sa pe un *server*.

Din nou formularul diferă în funcție de programul de navigare utilizat. În figurile 6, 7 și 8 sunt prezentate formularele afișate de programele de navigare *Opera* 7.02, *Netscape Navigator* 7.02 și *Internet Explorer* 6.0 *SP* 1.

```
<HTML>
  <HEAD>
    <TITLE>
      Încărcarea unui fișier
    </TITLE>
  </HEAD>
  <BODY>
    <FORM enctype=
      "multipart/form-
      data" action =
       "http://www.ginfo.
        ro/php/upload.php"
      method = "post">
      Alegeți fișierul:
      <BR>
      <INPUT name =
              "fisier"
        type = "file">
      <INPUT type =
       "hidden" name =
       "MAX_FILE_SIZE"
         value = 2457>
      <P>
      <INPUT type = "submit" value =
                        "Trimite fișierul!">
    </FORM>
  </BODY>
</HTML>
```

Figura 6: Opera

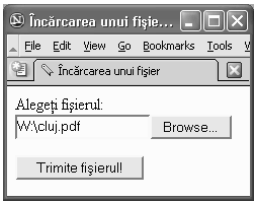

Figura 7: Netscape

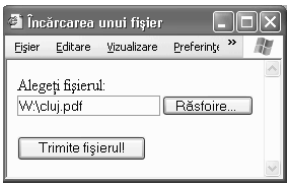

Figura 8: Internet Explorer

Butonul *Browse...* (*Choose*, *Răsfoire...* etc.) poate fi utilizat pentru alegerea unui fișier care urmează a fi încărcat. Apăsarea acestui buton va duce la apariția unei casete de dialog care permite alegerea fișierului.

Este posibilă specificarea fișierului precizând exact calea în caseta de text afișată.

URL-ul care este folosit ca valoare pentru atributul **action** al marcajului <**FORM**> trebuie să reprezinte documentul *PHP* care va realiza încărcarea.

Încărcarea fișierului va începe în momentul în care este apăsat butonul *Trimite fișierul!*.

Operația de încărcare

Așadar, formularul prin intermediul căruia este ales fișierul va fi prelucrat de către un *script PHP*.

Pentru a manipula fișierele poate fi utilizat tabloul `$_FILES`. Pentru *script*-ul care prelucrează formularul prezentat anterior, informațiile referitoare la fișierul încărcat vor fi accesate folosind variabila `$_FILES['fisier']` care este tot un vector.

Elementele acestui vector sunt prezentate în cele ce urmează:

- `$_FILES['fisier']['name']`: numele pe care îl are fișierul pe calculatorul utilizatorului;
- `$_FILES['fisier']['tmp_name']`: numele temporar pe care îl are fișierul pe *server* după încheierea procesului de încărcare;
- `$_FILES['fisier']['type']`: tipul fișierului în cazul în care programul de navigare furnizează această informație; un exemplu ar putea fi `image/gif`;
- `$_FILES['fisier']['size']`: dimensiunea fișierului, exprimată în octeți;
- `$_FILES['fisier']['error']`: un cod de eroare generat în urma efectuării încărcării.

Fișierul va fi stocat pe *server* în directorul care conține fișierele temporare.

O secvență de instrucțiuni *PHP* care poate fi folosită pentru a prelua fișierul încărcat și a-l muta în directorul dorit este următoarea:

```
<?PHP
  if (is_uploaded_file($_FILES['fisier']
                              ['tmp_name'])) {
    copy($_FILES['fisier']['tmp_name'],
                          "/directorul/dorit");
  }
  else{
    echo "Eroare de transfer " .
                   $_FILES['fisier']['name'];
  }
?>
```

O variantă alternativă este:

```
<?PHP
  move_uploaded_file($_FILES['fisier']
           ['tmp_name'],"/directorul/dorit");
?>
```

Coduri de eroare

Începând cu versiunea *PHP* 4.2.0, în urma efectuării unui transfer, sunt generate anumite coduri de eroare. Versiunea 4.3.0 introduce constante predefinite pentru fiecare dintre aceste coduri.

În funcție de codul de eroare generat în urma operației, trebuie realizate diferite acțiuni. Codul de eroare poate fi accesat prin intermediul elementului `['error']` al vectorului care păstrează informațiile despre fișier.

Codurile de eroare care pot fi generate sunt următoarele:

- `UPLOAD_ERR_OK` (`0`): nu a fost semnalată nici o eroare în timpul transferului; operația de încărcare s-a încheiat cu succes;
- `UPLOAD_ERR_INI` (`1`): dimensiunea fișierului este mai mare decât valoarea maximă permisă (această valoare este specificată în momentul configurării *server*-ului și poate fi schimbată modificând fișierul **PHP.INI**);
- `UPLOAD_ERR_FORM_SIZE` (`2`): dimensiunea fișierului este mai mare decât valoarea maximă indicată de directiva `MAX_FILE_SIZE` care este specificată în formular; această valoare este specificată prin intermediul unui câmp ascuns al cărui nume este `MAX_FILE_SIZE`;

- UPLOAD_ERR_PARTIAL (3): fișierul a fost încărcat doar parțial;
- UPLOAD_ERR_NO_FILE (4): nu a fost încărcat nici un fișier.

Încărcarea mai multor fișiere

Există posibilitatea de a încărca mai multe fișiere dacă sunt folosite nume diferite pentru marcajul <**INPUT**> utilizat pentru afișarea casetei de text și a butonului folosite pentru alegerea fișierului.

De asemenea, este posibil ca fișierele încărcate să fie organizate automat într-un vector. Pentru aceasta, la numele folosit trebuie adăugate caracterele '[]'. Un exemplu este prezentat în continuare:

```
<FORM action =
          "http://www.ginfo.ro/php/upload.php"
  method = "post"
  enctype = "multipart/form-data">
  Alegeți fișiele:<BR>
  <INPUT name = "fisiere[]" type = "file">
  <BR>
  <INPUT name = "fisiere[]" type = "file">
  <BR>
  <INPUT type = "submit" value =
                        "Trimite fișierele">
</FORM>
```

Să presupunem că am ales să încărcăm fișierele **F1.PDF** și **F2.PDF**. În acest caz, $_FILES['fisiere']['name'][0] va avea valoarea F1.PDF, în timp ce $_FILES['fisiere']['name'][1] va avea valoarea F2.PDF.

Dimensiunile celor două fișiere pot fi determinate folosind valorile $_FILES['fisiere']['size'][0], respectiv $_FILES['fisiere']['size'][1].

Includerea fișierelor

Deseori, mai multe pagini *Web* pe care le creăm trebuie să conțină secvențe identice. Evident, putem copia secvențele dintr-un fișier în altul, dar această variantă, pe lângă faptul că este incomodă, duce la un consum mărit de spațiu pe *server* și, în cazul în care secvențele sunt lungi sau sunt folosite în foarte multe pagini, problema este destul de importantă.

Interpretorul *PHP* permite păstrarea secvențelor comune în fișiere separate și accesarea acestora din cadrul altor documente *PHP*.

Pentru a accesa un astfel de fișier, poate fi utilizată directiva **include**. Practic, din punct de vedere logic, tot conținutul fișierului inclus este inserat în poziția în care apare directiva. Sintaxa acestei directive este **include** *fișier*;.

Să presupunem că avem un fișier numit **HELLO.TXT** cu următorul conținut:

```
<?PHP
  echo "Hello World!";
?>
```

În continuare vă prezentăm modul în care poate fi inclus acest fișier într-un document *PHP*. Pagina *Web* generată este prezentată în figura 9.

```
<HTML>
  <HEAD>
    <TITLE>
      Fișiere incluse
    </TITLE>
  </HEAD>
  <BODY>
    <?PHP
      include "HELLO.TXT";
    ?>
  </BODY>
</HTML>
```

Figura 9

Pentru ca fișierul inclus să fie interpretat, acesta trebuie obligatoriu să indice faptul că secvențele de text din cadrul său reprezintă instrucțiuni *PHP*. Așadar, prezența construcției <**?PHP** ... **?**> este necesară pentru ca textul să fie interpretat corect. Dacă fișierul **HELLO.TXT** ar conține doar secvența **echo** "Hello World!";, atunci aceasta nu ar fi interpretată ca fiind cod *PHP*, iar efectul ar fi cel din figura 10.

Figura 10

Aceasta se datorează faptului că în momentul includerii unui fișier se iese din modul *PHP* și tot textul este interpretat ca fiind cod *HTML*. După interpretarea textului din fișierul inclus se intră din nou în modul *PHP*.

În concluzie, folosirea celor două fișiere este echivalentă cu utilizarea următorului fișier:

```
<HTML>
  <HEAD>
    <TITLE>
      Fișiere incluse
    </TITLE>
  </HEAD>
  <BODY>
    <?PHP
      echo "Hello World!";
    ?>
  </BODY>
</HTML>
```

Includerea în cadrul funcțiilor

Dacă directiva de includere apare în cadrul unei funcții, atunci se consideră că toate instrucțiunile din cadrul fișierului inclus fac parte din corpul funcției. Așadar, domeniul de vizibilitate al variabilelor folosite în fișierul inclus este același cu domeniul pe care l-ar fi avut dacă ar fi apărut în cadrul funcției. Pentru a ilustra acest aspect vom considera un fișier **VAR.TXT** care conține secvența:

```
<?PHP
  $culoare = 'verde';
```

```
  $fruct = 'măr';
?>
```

și următorul document *PHP*:

```
<HTML>
  <HEAD>
    <TITLE>
      Includere în funcție
    </TITLE>
  </HEAD>
  <BODY>
    <?PHP
      function fructe(){
        global $culoare;
        include 'var.txt';
        echo "un $fruct $culoare<BR>";
      }
      fructe();
      echo "un $fruct $culoare<BR>";
    ?>
  </BODY>
</HTML>
```

După cum se poate vedea în figura 11, variabilele definite în cadrul fișierului inclus au domeniul de vizibilitate corespunzător funcției `fructe()`. Din aceste motive valoarea variabilei `$fruct` nu este disponibilă în afara funcției. Valoarea variabilei `$culoare` este disponibilă și în afara funcției, deoarece a fost declarată ca fiind o variabilă globală.

Figura 11

Așadar, instrucțiunea **echo** din cadrul funcției duce la afișarea mesajului `un măr verde`, deoarece ambele valori ale variabilelor sunt disponibile, iar instrucțiunea din afara funcției duce la afișarea mesajului `un verde`, deoarece, în afara funcției, variabila `$fruct` nu a fost definită anterior și este automat inițializată cu șirul vid.

Includeri prin HTTP

După cum ați observat, directiva **include** are ca parametru un șir de caractere care conține numele fișierului care va fi inclus. Acesta poate fi orice *URL* valid, așadar este posibilă includerea unui fișier aflat pe un alt *server*.

Un exemplu ar putea fi **include** `"http/www.ginfo.ro/php/ginfo.txt"`. O astfel de directivă funcționează doar dacă se folosește *PHP* 4.3.0 sau o versiune ulterioară.

Observație

Datorită faptului că secvențele din cadrul fișierului inclus sunt, din punct de vedere logic, inserate în poziția în care apare directiva **include**, trebuie să fim atenți la modul în care sunt folosite blocurile de instrucțiuni.

Să presupunem că avem un fișier **HELLO1.TXT** care conține secvența:

```
<?PHP
  echo "1. ";
  echo "Hello World!<BR>";
?>
```

și un fișier **HELLO2.TXT** cu următorul conținut:

```
<?PHP
  echo "2. ";
  echo "Hello World!<BR>";
?>
```

Dacă în cadrul unui document *PHP* folosim o secvență de tipul:

```
if ($ok)
  include "HELLO1.TXT";
else
  include "HELLO2.TXT";
```

atunci aceasta este echivalentă cu:

```
if ($ok)
  echo "1. ";
  echo "Hello World!<BR>";
else
  echo "2. ";
  echo "Hello World!<BR>";
```

Deși construcția din cadrul documentului *PHP* pare corectă, din cea echivalentă rezultă clar faptul că *script*-ul nu va funcționa așa cum ne-am aștepta.

Pentru a evita acest tip de eroare, construcția corectă este:

```
if ($ok){
  include "HELLO1.TXT";
}
else{
  include "HELLO2.TXT";
}
```

chiar dacă acoladele par inutile. Rolul lor devine evident dacă privim construcția echivalentă rezultată:

```
if ($ok){
  echo "1. ";
  echo "Hello World!<BR>";
}
else{
  echo "2. ";
  echo "Hello World!<BR>";
}
```

Includerea unui fișier o singură dată

Uneori, mai ales în situațiile în care se folosesc variabile pentru numele fișierelor incluse, este posibil ca anumite fișiere să fie incluse de mai multe ori. De obicei, fișierele incluse conțin definiții ale unor funcții sau clase, motiv pentru care este de dorit ca un astfel de fișier să nu fie inclus de mai multe ori.

Pentru ca un fișier să fie inclus o singură dată, se utilizează o directivă alternativă numit `include_once`. După cum rezultă din denumire, un fișier va fi inclus o singură dată. Cu alte cuvinte, în momentul în care se cere includerea unui fișier, se verifică dacă acesta a fost deja inclus și în

această situație directiva nu are nici un efect (fișierul nu este inclus încă o dată).

Alternative

Pe lângă directivele **include** și **include_once** există două alte directive a căror utilizare are un efect aproape echivalent. Acestea sunt **require** (echivalentă cu **include**) și **require_once** (echivalentă cu **include_once**).

Singura diferență dintre **include** și **require** (respectiv dintre **include_once** și **require_once**) este dată de modul în care este tratată situația în care fișierul care se dorește a fi inclus nu există sau nu este disponibil.

În cazul directivelor **include** și **include_once** este generat un *avertisment* și execuția *script*-ului continuă fără ca fișierul să fie inclus. Practic, din punct de vedere logic, este ignorată directiva.

În cazul directivelor **require** și **require_once** este generată o eroare fatală și execuția *script*-ului este întreruptă.

Așadar, în situațiile în care este absolut necesară existența fișierului, pentru a nu apărea efecte bizare, trebuie utilizată directiva **require**, respectiv **require_once**.

Starea conexiunii

În *PHP*, conexiunea poate avea una dintre cele trei stări posibile :

- `NORMAL` (`0`): starea normală de funcționare;
- `ABORTED` (`1`): utilizatorul care accesează pagina generată de *script*-ul aflat în execuție s-a deconectat (de obicei prin apăsarea butonului ***Stop*** al programului de navigare);
- `TIMEOUT` (`2`): a fost atinsă limita de timp impusă de *PHP*.

Pentru fiecare dintre cele trei stări există un *indicator* (*flag*) care este setat sau nu în funcție de starea conexiunii.

În funcție de *script*, este necesar sau nu ca execuția acestuia să se încheie în momentul în care este întreruptă conexiunea. De exemplu, în cazul în care *script*-ul actualizează o bază de date ar fi de dorit ca execuția sa să continue. În principiu, este indicat ca *script*-urile să își încheie execuția chiar dacă datele de ieșire nu vor mai fi trimise utilizatorului.

Pentru a decide dacă execuția *script*-ului se va încheia în momentul întreruperii conexiunii puteți (sau nu!) să apelați funcția `ignore_user_abort()`. Implicit, în momentul în care conexiunea este întreruptă, se întrerupe și execuția *script*-ului.

Este posibil (chiar recomandabil!) să fie definită o funcție care va fi executată înaintea întreruperii execuției *script*-ului. Aceasta poartă denumirea de ***funcție de dezactivare*** (*shutdown function*) și este definită la fel ca orice altă funcție.

Pentru a specifica faptul că o anumită funcție trebuie apelată înaintea întreruperii execuției, aceasta trebuie înregistrată ca fiind funcție de dezactivare. Aceasta se realizează printr-un apel al funcției `register_shutdown_function()`. În cadrul funcției se poate testa dacă execuția a fost determinată de întreruperea conexiunii prin apelarea funcției `connection_aborted()`. Aceasta va returna valoarea **TRUE** dacă a fost întreruptă conexiunea.

Funcția de dezactivare va fi apelată automat și în cazul în care este depășită limita de timp. În cadrul funcției se poate testa dacă un astfel de eveniment a avut loc prin apelarea funcției `connection_aborted()` care returnează valoarea **TRUE** în cazul depășirii acestei limite.

Există posibilitatea de a testa ambele indicatoare printr-un singur apel al funcției `connection_status()`. Funcția va returna un număr întreg ai cărui biți reprezintă indicatoarele. De exemplu, dacă sunt setate indicatoarele `ABORTED` și `TIMEOUT`, valoarea returnată va fi `3`.

Trebuie remarcat faptul că este posibil ca indicatorii `ABORTED` și `TIMEOUT` să fie ambii setați la un moment dat. Această situație poate apărea în cazul în care întreruperea conexiunii este ignorată. *PHP* va detecta faptul că a fost întreruptă conexiunea și va seta indicatorul corespunzător. Totuși, execuția *script*-ului va continua și este posibil să expire și limita de timp, moment în care este setat și indicatorul `TIMEOUT`.

Implicit, limita de timp este de 30 de secunde, dar ea poate fi modificată folosind funcția `set_time_limit()`.

Fișiere aflate la distanță

Așa cum am afirmat în secțiunea dedicată includerii fișierelor, începând cu versiunea *PHP* 4.3.0 este posibilă includerea de fișiere aflate pe alte *server*-e.

Evident, este posibilă utilizarea de fișiere aflate pe alte *server*-e și în alte scopuri. Practic, în aproape toate funcțiile care manipulează fișiere, pot fi folosite atât fișiere locale, cât și fișiere aflate la distanță.

În continuare vom prezenta modul în care pot fi citite date dintr-un fișier aflat pe *site*-ul revistei noastre:

```
<?PHP
  $file = fopen("http://www.ginfo.ro/
                              index.shtml","r");
  if (!$file) {
    echo "Fișierul nu poate fi accesat!\n";
    exit;
  }
  ... // citirea datelor
  fclose($file);
?>
```

Este posibilă și scrierea de date pe un *server* aflat la distanță. Fișierul va fi deschis folosind o secvență de tipul:

```
$file = fopen("http://www.ginfo.ro/
                              index.shtml","w");
if (!$file) {
  echo "Accesul nu este permis!\n";
  exit;
}
```

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.