Programare.org:

PHP

Manualul Programatorului

ver. 1.1 - Mai 2005

Prezentare

Acest e-book gratuit se adreseaza programatorilor sau tuturor celor care doresc sa invete PHP, incepind cu nivelul introductiv, pentru incepatori, dar si lucruri mai avansate. Toate notiunile vor fi introduse gradual si explicate. Nu este un inlocuitor pentru documentatia PHP-ului care este o prezentare completa a tuturor facilitatilor si functiilor limbajului, ci o metoda simpla si eficienta de a prezenta posibilitatile limbajului la lucru, incepind cu pagini si exemple simple.

Pentru intrebari legate de PHP dar si de alte probleme de programare va invitam pe site-ul nostru: programare.org, unde se poate intreba orice in forum. Ne vom stradui sa gasiti si raspunsuri, si in plus, articole si alte materiale interesante. Daca aveti ceva de spus despre programare, nu doar PHP va invitam pe site.

Daca aveti sugestii, corecturi, propuneri sau comentarii va rog sa ni le spuneti pe site si vom tine cont de ele in editiile urmatoare ale acestui e-book.

Unele materiale au fost luate din manualul oficial PHP, inclusiv cel in limba romana. In unele situatii au fost adaugate tagurile de cod PHP in altele nu, insa e clar ca este vorba despre cod PHP.

In acest e-book a fost lasat deoparte capitolul despre clase pentru ca ar trebui abordat mai pe larg si nu intotdeauna e folosit pentru site-uri simple.

Atentie la copierea codului prezentat aici ca este posibil ca fie inlocuite ghilimelele simple cu ghilimele de citare (ghilimele deschise + inchise), care creaza probleme la executia codului PHP. Sa nu uitat sa faceti o inlocuire a acestora daca e cazul.

Despre Programare.org

Programare.org este o noua comunitate virtuala a programatorilor romani care isi propune sa creeze un mediu de comunicare eficient pentru toti membrii sai prin forum de discutii, articole, stiri, webloguri dar si altele. Site-ul are in spate o echipa foarte mica de entuziasti care au nevoie de ajutorul dvs. Sintem foarte deschisi la propuneri si sugestii asa ca va invitam sa ne spuneti ce altceva doriti de la acest site.

Temele propuse pe site sint foarte variate: de la programare web, aplicatii sau baze de date, pina la stiri din domeniul IT sau chiar discutii generale. Stim ca exista si alte site-uri destinate programarii, insa unele sint specifice (PHP, VB, etc.), iar altele nu au reusit sa adune suport si au disparut.

La toti ne place sa gasim informatii pe Internet, insa acestea exista doar in masura in care lumea contribuie cu aceste informatii. Daca ati dat de o problema mai dificila va invit sa spuneti si la altii solutia gasita pentru a-i ajuta pe viitor, tot asa cum dvs. ati gasit sprijin cind ati avut nevoie. Sau chiar puteti promova business-ul dvs sau cauta colaborari sau colaboratori, insa toate in limita bunului simt (fara reclama exagerata), totul fiind gratuit.

In aceeasi idee se inscrie noua serie de e-books despre programare mai ales destinate incepatorilor, scrise in romaneste, simplu de inteles. Vizitati cit mai des site-ul pentru noutati sau alte e-books care au fost publicate sau se vor publica curind.

Pentru cei care doresc sa ne sprijine ii invit sa creeze un link catre programare.org. Acest lucru poate fi considerat plata pentru acest e-book gratuit daca il considerati util si v-a ajutat. Nu este obligatoriu ci doar o rugaminte.

Multumim.

Cuprins

1. Istoria PHP

Aceasta istorie nu este nici pe de parte completa. Totul a inceput in 1994-1995 cind Rasmus Lerdorf a creat o implementare C a PHP/LI pornind de la Perl pentru un site personal. PHP/LI insemna *Personal Home Page / Forms Interpreter*.

In 1997 PHP/LI a ajuns la versiunea 2.0 find rescris tot in C. Aproximativ 50.000 de site-uri au folosit acest limbaj, fiind un lucru deosebit, tinind cont ca a pornit ca proiect personal.

Tot in 1997 s-a oprit dezvoltarea la PHP/LI si a aparut PHP 3.0 rescris de Andi Gutmans si Zeev Suraski. Aceasta varianta seamana cu ceea ce stim noi din PHP. Tot acum a fost redenumuit simplu PHP de la numele recursiv *Hypertext Preprocessor*. Lansarea oficiala a PHP 3.0 a fost in iunie 1998.

Imediat dupa lansarea PHP 3.0 s-a inceput lucrul la urmatoarea versiune: 4.0. Noua versiune a fost introdusa la mijlocul lui 1999 si a poarta numele de 'Zend Engine' de la numele celor 2 creatori ai sai: Zeev si Andi.

Cei doi au fondat Zend Technologies (zend.com), compania care se ocupa de successul PHP-ului.

Pentru mai multe detalii vedeti php.net/history.

2. Ce este PHP?

Numele de Hypertext Preprocessor sau PHP nu spune prea multe despre ce este. Este un limbaj asemanator cu C sau Perl, cu variabile, constante, siruri, stringuri si alte tipuri de date, cu instructiuni de control si functii, cu obiecte si alte lucruri necesare oricarui limbaj, care interpreteaza cod scris pe server si returneaza cod HTML pentru pagini web.

Dupa cum stiti HTML este un format static, care este salvat in fisiere apelate din browsere pentru a fi vizualizate. Insa atunci cind se doreste crearea dinamica a fisierului HTML trebuie folosit pe server un limbaj de scripting: PHP, Perl, ASP sau altele. Din paginile respective se pot interoga baze de date, folosi informatii din alte fisiere sau chiar de pe alte site-uri, folosi emailuri sau alte date stocate in diverse modalitati, iar in final se construieste o pagina HTML pasata si vizualizata in browser. PHP este transparent pentru vizitatori, ceea ce rezulta fiind HTML.

Nu exista compilare in PHP, codul raminid sub forma de sursa pe server fiind interpretat la fiecare cerere. Se instaleaza librariile PHP-ului care stiu sa functioneze cu serverul web oricare e acela, iar la cererea unui fisier cu extensia .php (de obicei, pentru ca pot fi folosite si alte extensii) va fi chemat PHP-ul care interpreteaza codul specific, dintre tagurile PHP-ului, rezultind HTML. Iar in browserul clientului nu ajunge deloc cod PHP ci doar HTML. Asa ca parolele sau codul dvs. ramine de nemodificat sau nevazut pe server. Principiul este la fel si cu alte limbaje pentru Internet, server-side: ASP, Perl, ColdFusion, etc.

PHP se executa doar pe server si nu interactioneaza cu userului decit sub forma de HTML sau JavaScript.

Codul PHP este marcat cu citeva taguri speciale. Cel mai adesea este folosit:

```
<?php … cod PHP … ?>
```

Dar se poate scrie si doar <? … cod PHP … ?> sau
<script language="php"> … cod PHP … </script> sau chiar
<% … cod PHP … %> la fel cu ASP.

Codul PHP poate fi amestecat oriunde in pagina HTML, doar trebuie sa rezulte cod HTML, cu structura unui document HTML.

De exemplu:

```
<html>
<head>
<title>Prima pagina</title>
<?php … cod PHP … ?>
</head>
<body>
De afisat ceva
<?php … cod PHP … ?>
</body>
</html>
```

Si acum sa trecem la primele lucruri facute in PHP. Intii e vorba de comentarii. La fel ca in C se pot include comentarii care vor fi sarite la interpretarea paginii:

```
/*  acesta este un
comentariu multilinie */

// comentariu pe o singura linie
```

Primele instructiuni PHP vor fi prezentate dupa capitolul urmator despre instalare.

Cum se pot scrie aceste lucruri? Cu orice editor HTML sau chiar plain-text. Multi folosesc inclusiv Notepad sub Windows desi cu un editor specializat de HTML se poate lucra mai eficient si mai rapid. Multi programatori web folosesc Macromedia DreamWeaver, insa si alte editoare sint bune. Se poate folosi chiar Zend Studio de la Zend.com, firma din spatele PHP-ului.

3. Instalare PHP

PHP este un limbaj implementat pe mai multe platforme, de la Unix, Linux in diverse variante pina la Windows sau Mac OS X. De asemenea merge pe mai multe servere web instalate pe aceste platforme: Apache, iPlanet, IIS si altele.

Informatii complete despre instalare gasiti la http://php.net/manual/en/install.php .

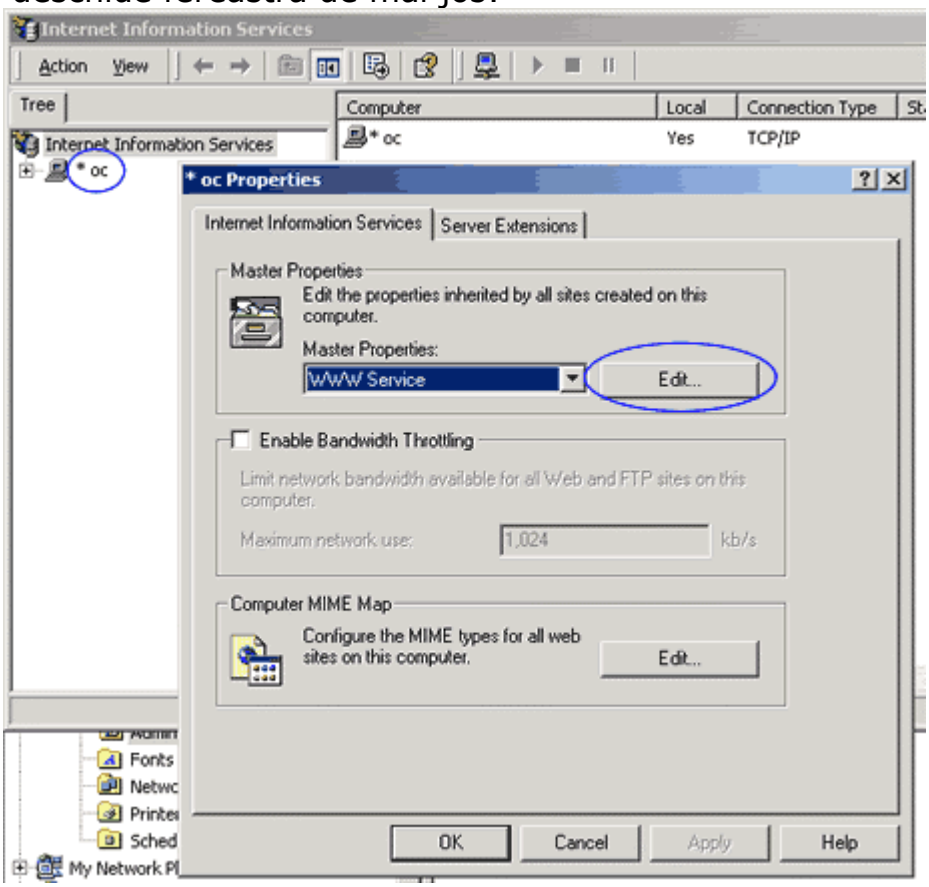
Pasii de urmat pentru Apache 2 Shared Module:

```
1.  gzip -d httpd-2_0_NN.tar.gz
2.  tar xvf httpd-2_0_NN.tar
3.  gunzip php-NN.tar.gz
4.  tar -xvf php-NN.tar
5.  cd httpd-2_0_NN
6.  ./configure --enable-so
7.  make
8.  make install
9.  cd ../php-NN
10. ./configure --with-apxs2=/usr/local/apache2/bin/apxs --
with-mysql
11. make
12. make install
13. Setup your php.ini
    cp php.ini-dist /usr/local/lib/php.ini
14. Edit your httpd.conf to load the PHP module.
    For PHP 4:
      LoadModule php4_module libexec/libphp4.so
    For PHP 5:
       LoadModule php5_module libexec/libphp5.so
15. AddType application/x-httpd-php .php .phtml
    AddType application/x-httpd-php-source .phps
16. /usr/local/apache2/bin/apachectl start
```

Multi useri instaleaza PHP pe sistemele lor folosind Windows cu serverul de web de la Microsoft: Internet Information Server (IIS). Procedura de instalare este descrisa la http://php.net/manual/en/install.windows.iis.php .

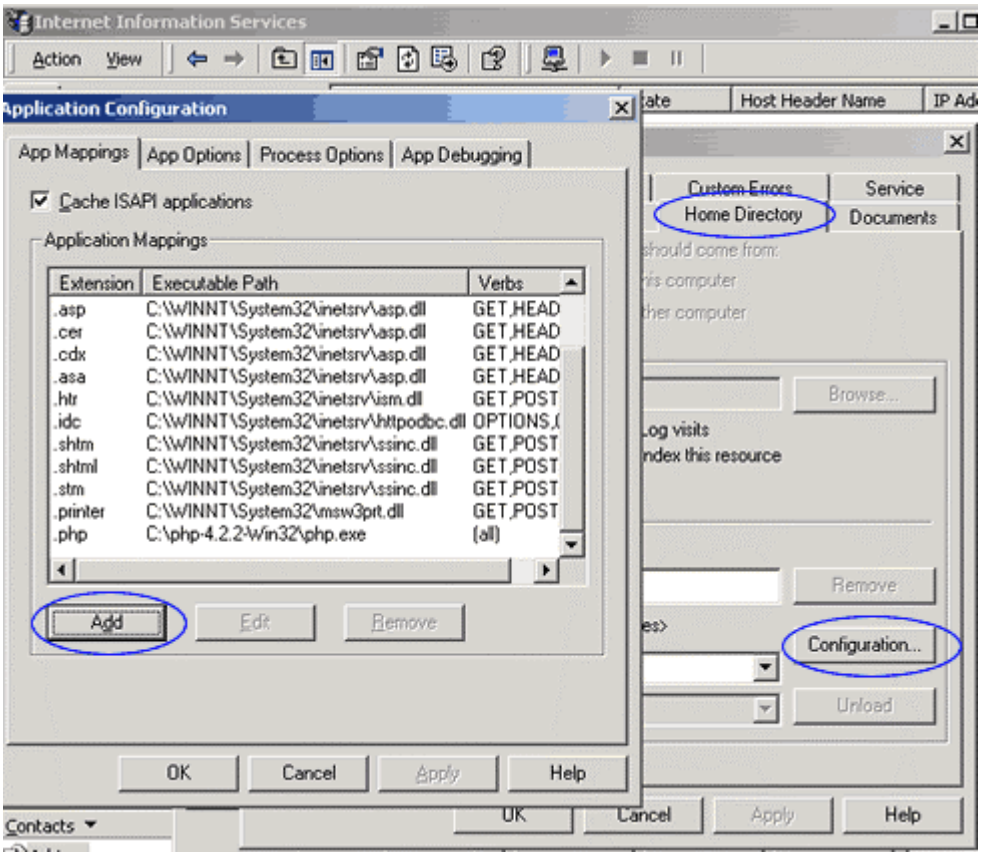
Exista doua solutii pentru a rula scripturi PHP pe Microsoft Windows: folosind ISAPI sau folosing CGI. Pentru prima varianta se folsoseste /sapi/php4isapi.dll de adaugat in lista filtrelor ISAPI pe IIS din Internet

Services Manager (Control Panel -> Administrative Tools). Cealalta varianta este mai simpla si se foloseste php.exe. Trebuie doar :
- copiat continutul arhivei PHP intr-un folder (c:\php)
- modificat php.ini si copiat in c:\Windows (sau c:\winnt)
- adaugat calea catre PHP in PATH (adaugati c:\php; in Control Panel -> System -> Advanced -> Environment Variables -> System Variables (jos) -> Path)
- in Control Panel -> Administrative Tools -> Internet Services Manager click dreapta pe Properties la numele computerului va deschide fereastra de mai jos:



Apoi deschideti Edit -> Home Directory -> Configuration unde dati click pe Add pentru a defini o noua mapare cu Executable avind calea catre php.exe si Extension = ”.php”.

Vedeti mai jos ferestrele specificate:



Dupa asta in folderele IIS-ului (c:\inetpub\wwwroot sau in alte foldere setate sub IIS) puteti folosi pagini PHP, pe linga ASP.

O prima pagina de test poate fi asta:

```
<?php phpinfo(); ?>
```

Doar aceasta linie simpla poate fi copiata intr-un fisier cu extensia .php (ex. test.php), copiat in root-ul serverului web sau alt folder (c:\inetpub\wwwroot sub IIS sau \htdocs sub Apache) si apoi chemat:

http://localhost/test.php

Ar trebui sa afiseze o pagina lunga cu setarile curente ale PHP-ului si modulele instalate.

Bravo! Tocmai ati scris prima dvs. pagina PHP. Desigur continuare o sa puna multe alte probleme intilnite in viata reala. Cu acest fel de pagina simpla puteti testa daca ceva functioneaza bine, daca aveti un modul instalat sau nu si alte lucruri utile.

4. Primele exemple simple

De fapt este al doilea, dupa exemplu anterior.
Cea mai simpla functie in PHP este echo() care afiseaza in browser parametrul pasat. Mai corect ar trebuie spus ca echo() nu este nici macar functie pentru ca pot lipsi parantezele, ci este o constructie PHP. Functia adevarata este print() care face la fel, dar care necesita un pic mai multe resurse decit echo() din moment ce este o functie ce returneaza o valoare (echo nu returneaza nimic).

Asa ca am putea scrie intr-un fisier cu extensia php (test2.php):

```
<html>
<head>
<title>Prima pagina</title>
<?php echo “<!-- comentariu scris din PHP -->”; ?>
</head>
<body>
De afisat ceva
<?php print(“<hr>\nLinia a 2-a”); ?>
</body>
</html>
```

Apoi chema:

http://localhost/test2.php sau http://127.0.01/test2.php

Sau chiar se poate executa din linia de comanda (command prompt):

c:\> php.exe c:\cale\test2.php

Dupa incarcarea paginii in browser incercati din meniul browserului View -> Source pentru viazualizarea sursei HTML a paginii. O sa vedeti doar cod HTML, nici urma de PHP. Asa cum am spus, PHP-ul doar se executa pe server si ceea ce rezulta este HTML simplu.

In exemplul de mai sus am folosit si *echo* si *print*, instructiuni PHP de afisare in browser, pentru a le vedea pe amindoua la lucru. De asemenea, am folosit ‘\n’ pentru a insera un rind nou in sursa HTML.

Dupa cum se vede toate instructiunile se termina cu ‘;’ (punct si virgula). Spatiile albe nu conteaza, iar o instructiune poate continua pe mai multe rinduri din moment ce o sa fie terminata cu ‘;’.

Alte functii simple care le puteti incerca la acest pas sint:

```
<?php
echo phpversion();
echo date(“r”);
echo rand(); ?>
```

Asta va afisa ceva de genul:

4.2.2Wed, 4 May 2005 18:26:43 -040013116

Dupa cum vedeti trebuie sa includeti cod HTML pentru ceva inteligibil, altfel totul o sa fie bagat unul in altul. De exemplu am putea avea:

```
<?php
echo “Versiunea PHP: “.phpversion().”<br>\n”.
“Data curenta: “ . date(“r”).”<br>\n”;
echo “Numar aleator: “.rand(); ?>
```

Asta o sa afiseze:

Versiunea PHP: 4.2.2
Data curenta: Wed, 4 May 2005 18:29:43 -0400
Numar aleator: 8147

Se poate vedea aici cum se pot folosi stringuri si concatena acestea. Si se poate vedea instructiune pe 2 linii.

5. Variabile predefinite

Exista citeva variabile globale de tip siruri asociative disponibile in toate paginile. Acestea sint: $_SERVER, $_GET, $_POST, $_SESSION, $_COOKIE, $GLOBALS, $_FILE, $_ENV si $_REQUEST.

Dupa cum se poate vedea variabilele in PHP sint precedate de '$'. O sa vedem in capitolul urmator mai multe lucruri despre variabilile obisnuite.

Despre $_ENV si $_REQUEST nu intru in detalii din moment ce nu sint foarte des folosite. $GLOBALS este pur si simplu un sir asociativ al tuturor variabilelor globale. Adica pentru o variabila obisnuita $var o sa putem scrie si $GLOBALS['var']. Dupa cum simplu se poate vedea un sir asociativ este un sir care in loc de indecsi numerici accepta stringuri. Toate acestea sint astfel de siruri, cu mentiunea ca pentru sirurile asociative nu exista indecsi numerici (sint doar de tip string nu si string si numerici). De exemplu putem avea:

```
<?php
$var=1;
echo $var."<br>\n";
echo $GLOBALS['var']."<br>\n";
echo $GLOBALS[0]."<br>\n"; ?>
```

Ceea ce va afisa:
1
1
PHP Notice: Undefined offset: 0 in xxxx on line yy

Adica ultima linie va genera o erorare, indexul numeric fiind inexistent.

$_SESSION va permite accesul la citeva informatii generale despre pagina chemata, browser, numele utilizatorului, calculatorul de unde se face conectarea si altele. Lista parametrilor disponibili sint:

'PHP_SELF'
: Adresa relativa a paginii curente (in care e folosit). De exemplu $_SERVER['PHP_SELF'] in pagina http://example.com/test.php/foo.bar va returna /test.php/foo.bar.

'SERVER_NAME'
: Numele hostingului sau serverului daca e disponibil.

'SERVER_SOFTWARE'

Server identification string, given in the headers when responding to requests.

'SERVER_PROTOCOL'
Numele protocolului folosit. De obicei este 'HTTP/1.0' sau 'HTTP/1.1';

'REQUEST_METHOD'
Felul chemarii paginii conform protocolului HTTP: 'GET', 'HEAD', 'POST', 'PUT'. De obicei este ‘GET’ sau ‘POST’

'QUERY_STRING'
Parametrii paginii folositi in adresa de chemare a paginii (URL)

'DOCUMENT_ROOT'
Folderul radacina a web serverului asa cum e definit in php.ini

'HTTP_ACCEPT'
Continutul cimpului Accept: din headerul cererii. Se poate crea raspunsul in functie de valorile acceptate in browser.

'HTTP_ACCEPT_CHARSET'
Continutul cimpului Accept-Charset: din headerul cererii care poate fi folosit pentru raspunsuri in alte limbi cu alt set de caractere.
Examplu: 'iso-8859-1,*,utf-8'.

'HTTP_ACCEPT_ENCODING'
Valoarea cimpului Accept-Encoding: din headerul cererii, daca exista.
Examplu: 'gzip' pentru raspunsuri comprimate, daca browserul permite.

'HTTP_ACCEPT_LANGUAGE'
Continutul cimpului Accept-Language: din headerul setat pe calculatorul client in functie de care se poate folosi diverse limbi.
Example: 'en'.

'HTTP_CONNECTION'
Continutul cimpului Connection: din headerul cererii, daca este setat.
Example: 'Keep-Alive'.

'HTTP_HOST'
Contine valoarea cimpului Host: din header cu domeniul de unde se face cererea, daca este setat.

'HTTP_REFERER'
Pagina anteriara cererii curente (de unde vine vizitatorul). Aceasta valoare nu intotdeauna e setata asa ca nu va puteti baza 100% pe ea (nu e sigura).

'HTTP_USER_AGENT'
Continutul cimpului User-Agent: din header cu informatii despre browser.Se poate folosi pentru a raspunde diferit pentru diferite browsere. Un exemplu tipic este: Mozilla/4.5 [en] (X11; U; Linux

2.2.9 i586). Pentru aceleasi lucruri va puteti uita si la functie **get_browser()**.

'REMOTE_ADDR'
Adresa IP a vizitatorului (celui care a deschis pagina).

'REMOTE_HOST'
Numele calculatorului de unde este deschisa pagina (a vizitatorului). Vedeti de asemenea **gethostbyaddr()**.

'REMOTE_PORT'
Portul folosi pentru vizualizarea paginii.

'SCRIPT_FILENAME'
Calea absoluta a paginii executate.

'SERVER_ADMIN'
Valoarea directivei SERVER_ADMIN pentru Apache (folosita la hostinguri virtuale).

'SERVER_PORT'
Portul folosit de server. De obicei 80.

'PATH_TRANSLATED'
Calea absoluta pe server la fisierul executat, daca serverul permite.

'SCRIPT_NAME'
Numele paginii exeutate.

'REQUEST_URI'
Calea relativa de acces a paginii.

'PHP_AUTH_USER'
Numele userului daca pagina cere autentificare Apache.

'PHP_AUTH_PW'
Parola introdusa la autentificarea Apache.

De exemplu puteti incerca:

```
<?php
echo $_SERVER['PHP_SELF']."<br>\n";
echo $_SERVER['HTTP_USER_AGENT']."<br>\n";
echo $_SERVER['REMOTE_ADDR']."<br>\n";
?>
```

Ceea ce la mine afiseaza:

/test3.php
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; .NET CLR 1.0.3705)
192.168.123.1

$_SESSION este un alt sir disponibil global cu variabile care tin valori intre diferite apeluri de pagini, pastrind valoarea lor pe toata durata vizitei unui site. O sa intram mai tare in detalii mai tirziu despre sesiune si functiile necesare pentru pornirea sau oprirea unei sesiuni. Folosirea variabilelor pastrate in $_SESSION este foarte simpla:

```
$_SESSION[‘var’]=1;
echo $_SESSION[‘var’];
```

Insa sint si alte detalii despre sesiune care vor fi prezentate dupa introducerea altor notiuni, intr-un capitol separat.

$_COOKIE pastreaza sau seteaza valori ale unor variabile pe calculatorul client in fisiere mici permanente sau doar pe durata cit e deschis browserul. Aceste valori sint pasate in mod invizibil in headerul (antetul) cererilor catre server sau ale raspunsurilor si pot contine orice valori doriti. Se poate specifica un timp cit cookie-ul este pastrat pe calculator, insa trebuie tinut cont si de faptul ca aceste informatii pot fi sterse. Vom da si alte informatii despre cookies mai tirziu.

Pe linga $_COOKIE se poate folosi si functia setcookie().

De exemplu:

```
<?php
$_COOKIE['x'] =555;
setcookie("xx",333);
print_r($_COOKIE); ?>
```

Va afisa:

Array ([xx] => 333 [x] => 555)

Am introdus aici o noua functie: print_r() care afiseaza (de obicei pentru debug) toate valorile unui sir cu tot cu indecsii numerici sau cheile de tip string.

$_FILE este un sir folosit pentru upload de fisiere. Are citeva chei predefinite cu care se apeleaza parametrii fisierelor uploadate (nume temporar, marime, etc.). Voi da mai tirziu citeva exemple de folosire.

Am lasat la urma cele mai des folosite variabile predefinite: $_GET si $_POST. Cu acestia puteti avea valorile parametrii pasati paginii atit in adresa de chemare a paginii (URL) cind este folosit HTTP GET, cit si in

valori pasate de obicei din forme HTML prin HTTP POST. De exemplu o sa vedeti des:

```
http://server.com/pagina.php?param1=1&param2=2
```

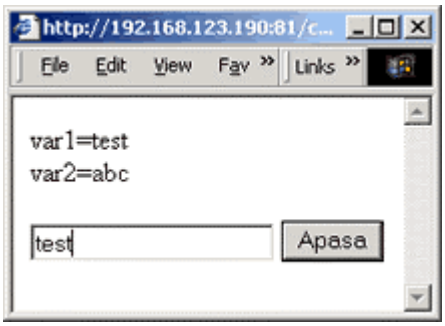
Pentru a obtine valorile din ‘param1’ si ‘param2’ respectiv ‘1’ si ‘2’ o sa putem scrie $_GET[‘param1’] sau $_GET[‘param2’]. De exemplu:

```
echo $_GET[‘param1’].” - “.$_GET[‘param2’];
```

Va afisa: 1 - 2

Pentru a testa $_POST voi da un mic exemplu care apoi va fi aprofundat in capitolul de lucru cu formele HTML. Sa zicem ca avem urmatoarea pagina (test4.php):

```
<html><body>
<?php echo
“var1=”.$_POST[‘var1’].”<br>var2=”.$_POST[‘var2’]; ?>
<form method=”post” action=”test4.php”>
<input type=”hidden” name=”var2” value=”abc”>
<input type=”text” name=”var1”>
<input type=”submit” name=”submit” value=”Apasa”>
</form>
</body></html>
```



Initial se va invarca doar text-box-ul cu textul de introdus (‘test’ in acest caz) si butonul de apasat. La apasarea butonului sint 2 parametri care sint pasati paginii prin HTTP POST: cimpul hidden (ascuns) numit ‘var2’ care are valoarea ‘abc’ si textul introdus de noi (poate fi orice). Dupa apasarea butonului pagina se reincarca si vor fi afisati parametrii prin codul PHP. Exista acolo un test (isset()) care verifica daca e vorba de prima incarcare a paginii sau o re-incarcare dupa pasare parametrilor. Numai in cazul al doilea va fi afisat textul cu valorile parametrilor.

Aici am dorit doar sa arat un exemplu simplu de folosire a tagului <form> din HTML cu pasarea parametrilor folosind metoda POST (method=”post”) si de preluare a valorilor pasate in codul PHP. Mai tirziu o sa intram in alte detalii despre lucrul cu forme.

Toate aceste forme au cite o alternativa mai veche si mai rar folosita:

- $_SERVER se poate scrie si $HTTP_SERVER_VARS
- $_POST se poate scrie si $HTTP_POST_VARS
- $_GET se poate scrie si $HTTP_GET_VARS
- $_COOKIE se paote scrie si $HTTP_COOKIE_VARS
- $_FILES se poate scrie si $HTTP_POST_FILES

Exista totusi o diferenta intre noile forme si cele vechi: cele noi sint automat globale si pot fi folosite peste tot, inclusiv in functii.

Cam acestea deocamdata despre variabilele predefinite, urmind ca alte detalii sau exemple de folosire sa fie aratate pe parcurs.

6. Variabile, constante, operatori

Sper ca nu am incurcat lumea prea tare prezentind intii variabilele predefinite si abia apoi cele normale, constante sau functiile definite de utilizator.

Asa cum am aratat deja orice nume de variabila incepe cu '$'. Este o conventie PHP si este necesara intotdeauna. Variabilele nu se definesc si nu au un tip propriu-zis (adica tipul este determinat in functie de valoarea tinuta si poate fi schimbat). Se poate scrie direct:

```
$var1=1;
$var2=”un string”;
$var3=true;
$var1=$var1+1;
```

Nu sint valide nume de variabile care incep cu altceva decit litere sau '_' (de exemplu nu e valid $1a), nici cele care au diverse alte caractere (#,%, spatiu, etc.). Sint lucruri de bun-simt la fel ca in alte limbaje.

Dupa declaratiile si initializarile de mai sus se poate schimba tipul 'variabilei', desi nu e chiar un tip:

```
$var1=”convertit la string”;  // era valoarea numerica
```

Exista citeva functii pentru a determina tipul variabilei: is_array(), is_bool(), is_float(), is_int(), is_integer(), is_long(), is_real(), is_string(), is_object(), is_null(), is_numeric() si altele citeva. Toate acestea returneaza TRUE daca valoarea variabilei este una din aceste valori.

Exemple:
is_string($var1) => TRUE
is_integer($var1) => FALSE

Exista apoi citeva functii de lucru cu variabile pentru a testa valorile ei sau elibera memoria variabilei:

- isset() si empty() verifica existenta unei variabile
- unset() – elibereaza datele variabilelor.

Exemplu de eliberare a unei variabile pentru a anu ajunge sa fie folosita mai tirziu in alta parte a paginii (eventual chiar de hackeri):

```
unset($var1);
```

Se poate apoi obtine tipul unei variabile, dar si seta cu ajutorul functiilor gettype() si settype(). Valorile acceptate sint:

- "boolean" (sau, până la PHP 4.2.0, "bool")
- "integer" (sau, până la PHP 4.2.0, "int")
- "float" (disponibil începând cu PHP 4.2.0, pentru versiunile anterioare se foloseşte varianta învechită "double")
- "string"
- "array"
- "object"
- "null" (începând cu PHP 4)

De exemplu:

```
$var1=1;
echo gettype($var1);
settype($var1,float);
```

Constantele se definesc cu define() si exista o conventie de notare (neobligatorie ci doar recomandata) de a se folosi litere mari pentru numele ei. La constante nu se foloseste '$'. Evident nu se poate schimba valoarea, dar pot fi folosite in calcule (de partea dreapta a semnului '='):

```
define("CONSTANTA1",1);
define("EROARE","A aparut o eroare!");
```

Apoi o sa poata fi folosite simplu:

```
$var1=CONSTANTA1+10;
echo EROARE;
```

Pentru constante exista functia constant() care returneaza valoarea ei, adica

```
echo EROARE;
echo constant("EROARE");
```

sint perfect la fel. Deci inutila a 2-a varianta, aproape intotdeauna, decit daca numele constantei este generat dinamic:

```
$i=1;
echo constant("CONSTANTA".$i) ;
```

Operatorii se impart in mai multe categorii, la fel ca in multe alte limbaje:

- Matematici: +, -, *, /, % (modulo)
  Exemple:
  ```
  $a = $b + $c;
  echo $a*10;
  ```

- Atribuire: =
  Exemple:
  ```
  $a=3;
  $a=($b = 3) + 5;   // $a=8 si $b=3
  ```

- Pe biti: & (AND), | (OR), ^ (XOR), ~ (NOT), << (rotire la stinga), >> (rotire la dreapta)
  Exemple:
  ```
  $a = $a | $b;
  $x = $y ^ 8;
  $b = $b << 1;
  ```

- De comparatie: >, <, <=, >=, == (egalitate), === (identitate), <> (sau !=), !== (ne-identic).
  Exemple:
  ```
  $a > $b, $a <> $b sau $a!= $b,
  $a === $b; // $a este egal cu $b si au acelasi tip
  ```

- Incrementare / decrementare: ++, --
  Ca si in C exista pre-incrementare (++$a, --$a) si post-incrementare ($a++, $a--)

- Logici: AND sau &&, OR sau ||, XOR, ! (=not)
  Exemple:
  ```
  ($a<1) OR ($a>10) este echivalent cu ($a<1) || ($a>10)
  ! $a;   // notati diferenta fata de ~ = negare pe biti, nu logica
  ```

- String: . (punct) = concatenare de stringuri
  Exemplu:
  ```
  $a = “inceput”;
  echo $a.” un string”;
  ```

- Siruri: + (= reuniune de siruri), == (egalitate), === (identitate, au aceleasi elemente in aceeasi ordine), <> (ne-egalitate), !== (ne-identic)

Exemple de folosire o sa fie date la capitolul siruri.

- Controlul erorilor: @
  Nu raporteaza erorile in instructiunea folosita, oricare ar fi aceste erori. Se foloseste pentru a sari peste mesajele de erorare standard si testarea valorilor returnate dupa terminarea instructiunii, eventual cu afisarea unui mesaj de eroare dat de programator.
  Exemplu: @$a=10/@b; // nu afiseaza "PHP Warning: Division by zero"

- Operatorul tertiar: conditie?adevarat:fals
  echo $i==1?"i este 1":"i nu este 1";

Sint unii operatori care se refera la notiuni ne-explicate inca aici (siruri sau stringuri), insa cred ca sint lucruri simple, usor de inteles. Pentru detalii vedeti si capitolele referitoare la acestea.

7. Instructiuni PHP

La fel ca multe alte limbaje si PHP are citeva instructiuni Acestea sint: if – else – elseif, while, do..while, for, foreach, break, continue, switch, return, require(), include(), require_once(), include_once(). Sa le vedem pe rind.

if – else – elseif
Testeaza conditia data si in functie de asta continua cu o ramura a instructiunii sau cu alta, daca exista alternativa. Altfel nu executa nimic.

Exemple:

```
if($a>1) echo “var. a mai mare decit 1”;
// este afisat doar daca $a>1

if($a>1) {
echo “var. a mai mare decit 1”;
$a=1;}
//ex. de instructiune compusa, folosind { } pentru grupare

if($a>1)       // exemplu de alternativa cu else
     echo “var. a mai mare decit 1”;
else
echo “var. a mai mica sau egal cu 1”;

if ($a > $b) {
    echo "a mai mare decit b";
} elseif ($a == $b) {
    echo "a egal cu b";
} else {
    echo "a mai mic decit b";
}
```

Ultimul exemplu combina toate variantele, inclusiv *elseif*.
Exista si o varianta alternativa:

```
if($a>1):
     echo “var. a mai mare decit 1”;
endif;
```

Aici se foloseste ‘:’ si *endif*. (Insa e mai rar folosita)

while
Forma ei este
while (conditie) instructiune;
Sau poate fi
while (conditie): instructiune; endwhile;

De obicei prima forma este folosita. Aceasta instructiune va repeta instructiunile atita timp cit conditia este adevarata:

```
$i=0;
while($i<5)
    {echo ++$i.” “;}
```

Va afisa: 1 2 3 4 5

do … while
Este foarte asemanatoare cu *while* doar ca o sa avem conditia de testat la sfirsitul ciclului. Ciclul se termina cind conditia nu mai este adevarata:

```
$i = 0;
do {
    echo $i;
} while ($i > 0);  // va parcurge ciclul doar o data
```

for()
Forma instructiunii este:
for(initializare;conditie;incrementare) instructiune;

De exemplu putem avea:

```
for($i=0;$i<10;$i++) {echo $i.” “;}
```

foreach()
Aceasta instructiune este asemanatoare cu *for()* doar ca parcurge siruri. Are 2 forme diferite:
foreach(sir as $variabila) instructiune;
foreach(sir as cheie => $variabila) instructiune;

Cu prima se poate accesa doar valoarea unui element (in $variabila), iar cu a doua se poate accesa atit cheia cit si valoarea unui element (in $cheie, respectiv $variabila).

Exemplu:

```
$a=array(1,2,3);
foreach($a as $val) echo $val.” “;
foreach($a as $cheie => $val) {
echo “\n<br>cheia: “.$cheie.” – valoare: ”.$val”; }
```

break
Termina executia la una din instructiunile for, foreach while, do..while sau switch.
Exemplu:

```
for ($i = 1; ; $i++) {
    if ($i > 10) {
        break;
    }
    echo $i;
}
```

Dupa cum vedeti conditia de oprire in for() lipseste insa este testat in corpul instructiuni. Odata ce $i > 10 o sa termine ciclarea datorita instructiunii break.

continue
Aceasta instructiune este folosita in instructiunile repetitive si diferita fata de *break* deoarece sare peste partea ramasa din ciclu la urmatorul element.

De exemplu:

```
for ($i = 0; $i < 5; ++$i) {
      if ($i == 2)
          continue
      print "$i\n";
  }
```

pur si simplu o sa sara peste $i egal cu 2.

switch
Instructiunea switch combina mai multe ramuri ale instructiunii if-elseif-else intr-o singura instructiune, folosita pentru selectia unor valori dintre mai multe optiuni:

```
switch ($i) {
case 0:
```

```
    echo "i este 0";
    break;
case 1:
    echo "i este 1";
    break;
case 2:
    echo "i este 2";
    break;
default:
    echo "i nu este 0, 1 or 2";
}
```

In functie de valoarea lui $i o sa intre pe o ramura sau alta a instructiunii. In caz ca nici un test nu este adevarat o sa intre pe ultima ramura (default). Aceasta ramura default poate lipsi, caz in care nu se executa nici o ramura. Dupa cum vedeti switch are nevoie de break pentru terminarea unei ramuri cind conditia este adevarata.

return
Este folosita in functii pentru intoarcerea controlului executiei programului cu sau fara returnarea unei valori. O vom vedea in detalii la capitolul functii.

include() si *require()*
Amindoua instructiunile include fisiere externe in cadrul paginii curente. La intinirea lor fisierul specificat este inclus complet in locul instructiunii. Diferenta intre ele este faptul ca dac afisierul de inclus lipseste sau da eroare include() va da o avertizare, iar require() opreste executia paginii cu ‘fatal error’.

Exemple:

```
include 'file.php';
include 'http://www.example.com/file.php?foo=1&bar=2';
require ‘file2.php’;
require(‘file3.php’);
```

Dupa cum se poate observa se pot include fisiere locale dar si chiar de pe Internet, folosing protocolul HTTP, eventual chiar cu parametri. Parantezele se pot folosi sau nu (sint optionale).

include_once() si *require_once()*
Diferenta intre acestea si cele anterioare este faptul ca orice includere se face o singura data, chiar daca fisierul de inclus apare de mai multe

ori, eventual chiar in diverse fisiere incluse. Este utila pentru evitarea definirii de 2 sau mai multe ori a aceleasi structuri, ceea ce duce la eroare.

Folosind includerile deseori este foarte convenabil de pastrat intr-o pagina separata diversi parametri globali ai paginilor sau cod comun, necesar in multe situatii. De exemplu, sa zicem ca avem 2 pagini – config.php si pagina.php:

```
<?php  //config.php
define(“NRMAX”,10);
$pas=2;
?>
```

```
<?php  // pagina.php
include ‘config.php’;
for($i=0;$i<NRMAX;$i=$i+$pas) echo $i.” “;
?>
```

Apelul paginii pagina.php va afisa: 0 2 4 6 8

8. Stringuri

Un string este la fel ca in alte limbaje o serie de caractere delimitate de apostroafe (‘) sau ghilimele (“) sau sintaxa heredoc. Exemple:

```
$a=’un string simplu’;
$b=”alt string”;
$c=<<<EOD
un string
pe mai multe
rinduri.
EOD;
```

Nu se poate combina inceputul stringului cu apostroafe iar sfirsitul cu ghilimele, sau invers. Exemplu incorect: ‘abc” sau “abc’.

Ultimul exemplu este cel numit heredoc, in care puteti include stringuri pe mai multe rinduri. ‘EOD’ poate fi inlocuit cu orice altceva doriti, doar sa fie repetat la sfirsitul stringului.

Sau se pot construi stringuri folosind si operatorul de concatenare punct (.).
Atentie: nu este ‘+’ ci ‘.’

```
$a=”un string”.” cu concatenare”;
$b=’alt string’.” cu concatenare”.
   “chiar pe mai multe rinduri”;
```

Cind se folosesc ghilimelele se pot include si caractere special: \n (LN sau linefeed), \r (CR sau carriage return), \t (tab orizontal), \\ (backslash), \” (caracterul ghilimele) si \$ (dolar). Exemplu:

```
$a=”un string cu ghilimele(\”)\nrindul 2 - \$”;
```

Pentru a obtine caracterul de la pozitia n dintr-un string puteti folosi $a[$n] sau $a{$n}. In exemplul de mai sus $a[0] sau $a{0} va fi ‘u’ (indexul incepe de la 0).

La PHP puteti folosi intr-un string delimitat cu ghilimele (“) si nume de variabile care vor fi inlocuite automat cu valoarea lor. De exemplu:

```
$a=”123”;
$b=” valoarea este: $a”; // va returna “valoarea este: 123”
```

Exista insa situatii cind se doreste afisarea unui alt string imediat concatenat la variabila data. De exemplu mai sus “valoarea este:

123a”. Pentru asta nu se poate scrie “valoarea este: $aa” pentru ca s-ar face inlocuirea cu variabila inexistenta $aa. Pentru aceste cazuri se poate scrie:

```
$b=” valoarea este: {$a}a”; // va returna “valoarea este: 123a”
```

Aceasta cred ca este varianta preferata pentru a inlocui valoarea unei variabile in stringuri date, mai exact folosind “{ }” pentru delimitarea numelui variabilei, in afara operatorului de concatenare: $b=' valoarea este: '.$a.'a';
De retinut ca acestea nu functioneaza cu delimitatori apostroafe (‘):

```
$b=’valoarea este: {$a}a’; // nu functioneaza
```

Convertirea unor valori numerice la string sau invers, de la string la valori numerice se poate face foarte simplu cu citeva functii: strval(), intval(), floatval() sau chiar operatiuni de cast. Exemple:

```
$a=(string)1;
$a=strval(1); // sint echivalente

$a=”1.2”
$i=intval($a);
$f=floatval($a);
```

Trebuie putina atentie la operatiuni care cuprind diverse tipuri de date. De exemplu:

```
echo "abc "+ 5;
echo "abc ". 5;
echo 5 + “2 abc”;
```

Prima returneaza 5 ca valoare numerica, deoarece ‘+’ este operatiune matematica, iar “abc” este convertit la numeric (intval(“abc”) => 0 la fel ca (string)”abc”). Atunci vom avea 0 + 5 = 5.

A doua va folosi concatenarea stringurilor si va afisa “abc 5”.

Ultima varianta va afisa 7, deoarece intval(“2 abc”) = 2

Exista apoi o multitudine de functii care se aplica stringurilor. Dintre cele mai folosite amintesc:

- strlen() – returneaza lungimea unui string

- trim(), ltrim(), rtrim() – elimina spatiile albe, eventual doar din stinga sau din dreapta
- strpos() – gaseste pozitia unui string in altul, eventual pornind de la o pozitie data
- strrpos() – gaseste pozitia unui string in altul pornind de la sfirsit, eventual pornind de la o pozitie data
- substr() – returneaza un substring de la o pozitie data, de lungime data
- explode() – imparte un string in substringuri bazat pe un string dat. Returneaza un sir de stringuri.
- implode() – inversul lui explode() -undeste un sit de stringuri intr-un singur string.
- strcmp() – compara 2 stringuri
- strtolower() – converteste un string la litere mici
- strtoupper() – converteste un string la litere mari
- substr_count() – numara aparitiile unui substring intr-un string dat
- substr_replace() – inlocuieste o parte a unui string cu alt string dat
- str_replace() – inlocuieste toate aparitiile cu altceva dat (poate fi string sau sir)
- wordwrap() – formateaza un string ca HTML de lungime data

Multe din aceste functii au variante pentru stringuri case-sensitive sau nu (diferenta intre litere mici/mari sau nu). Exemplu: str_ireplace() similar cu str_replace doar ca e case-insensitive, stripos() este case-insensitive strpos(), stristr() case-insensitive strstr(), s.a.m.d.

Pentru o lista completa a functiilor referitoare la stringuri vedeti http://www.php.net/manual/en/ref.strings.php .

O sa dau aici un mic exemplu de pagina care foloseste unele din aceste functii:

```
<?php
$a="un string";

echo "\n<br>\$a='".$a."'".
"\n<br>Lungime: ".strlen($a).
"\n<br>Caractere mari: ".strtoupper($a).
"\n<br>Pozitia 'st': ".strpos($a,"st").
"\n<br>Inlocuieste 'n' si 't' cu '_':
".str_replace(array("n","t"),"_",$a).
"\n<br>Pozitia 'st': ".strpos($a,"st").
"\n<br>Impartit in sir de stringuri: ";
```

```
$ar=explode(" ",$a);
print_r($ar);

echo "\n<br>String reunit din sir: ".implode(" ",$ar);

// Exemple pentru substr()

$rest = substr("abcdef", 1);    // returneaza "bcdef"
$rest = substr("abcdef", 1, 3); // returneaza "bcd"
$rest = substr("abcdef", 0, 4); // returneaza "abcd"
$rest = substr("abcdef", 0, 8); // returneaza "abcdef"

// Accesare substring cu {}
$string = 'abcdef';
$rest = $string{0};               // returneaza a
$rest = $string[3];               // returneaza d

// Pot fi si indici negativi
$rest = substr("abcdef", -1);    // returneaza "f"
$rest = substr("abcdef", -2);    // returneaza "ef"
$rest = substr("abcdef", -3, 1); // returneaza "d"
$rest = substr("abcdef", 0, -1);  // returneaza "abcde"
$rest = substr("abcdef", 2, -1);  // returneaza "cde"
$rest = substr("abcdef", 4, -4);  // returneaza ""
$rest = substr("abcdef", -3, -1); // returneaza "de"
?>
```

Un alt set de functii referitoare la stringuri lucreaza cu *expresii regulare (regex)*. Acestea sint functii care lucreaza pe baza unui model (pattern) de cautare codificata a unor parti din stringul initial. Nu o sa intru in detalii despre modelele de cautare pentru ca pot fi foarte complexe si doar despre acest subiect se pot scrie carti intregi, dar o sa enumar citeva elemente de definire a unui model:
^ = inceputul modelului
$ = sfirsitul modelului
. (punct) = orice caracter mai putin linie noua
[ ] = interval de cifre sau litere
\ = caracterul escape
| = solutii alternative (SAU)
( ) = sub-model
\d = orice cifra
\s = orice spatiu alb
\w = un cuvint

Functiile pentru stringuri care folosesc expresii regulare incep cu `preg_':

- preg_replace() – inlocuieste un string cu altul
- preg_split() – imparte un string intr-un sir de stringuri bazat pe un model
- preg_match() si preg_match_all() - regaseste un substring intr-un string dat bazat pe un model. Diferenta e ca prima functie se opreste la primul substring gasit, iar a doua le returneaza pe toate

Exemple:

```
<?php
$string = "April 15, 2003";
$pattern = "/(\w+) (\d+), (\d+)/i";
$replacement = "\$1 \$3";
echo preg_replace($pattern, $replacement, $string);
// afiseaza ‘April 2003’

/* Converteste HTML la text */
$search = array (
  "'<script[^>]*?>.*?</script>'si",// Elimina javascript
  "'<[\/\!]*?[^<>]*?>'si",           // Elimina taguri HTML
  "'([\r\n])[\s]+'",                 // Elimina spatii albe
  "'&(quot|#34);'i",           // Inlocuieste entitati HTML
  "'&(amp|#38);'i",
  "'&(lt|#60);'i",
  "'&(gt|#62);'i",
  "'&(nbsp|#160);'i",
  "'&(iexcl|#161);'i",
  "'&(cent|#162);'i",
  "'&(pound|#163);'i",
  "'&(copy|#169);'i",
  "'&#(\d+);'e");                    // Evalueaza ca PHP

$replace = array ("", "", "\\1", "\"", "&", "<", ">", " ",
     chr(161), chr(162), chr(163), chr(169), "chr(\\1)");

$text = preg_replace($search, $replace, $document);
//  $document trebuie initializat
```

```
/*Ex.: extragerea adresei de email dintr-un text oarecare */
$string = "123 abc ceva@undeva.domeniu-10.com #$% 321 xyz";
$pattern =
  "/([a-z][a-z0-9_.-\/]*@[^\s\"\)\?<>]+\.[a-z]{2,6})/i";
preg_match($pattern, $string, $email);
```

```
echo $email[0];// afiseaza doar 'ceva@undeva.domeniu-10.com'
?>
```

Expresiile regulare sint un instrument foarte puternic de cautare in stringuri, insa necesita exercitiu si o cunoastere destul de aprofundata.

Exista apoi functii care se refera la caractere UNICODE, cu suport pentru caractere internationale. Acestea incep de regula cu "mb_" si sint similare cu cele pentru stringuri normale:

- mb_strtolower(),mb_strtoupper() – converteste in litere mici / mari
- mb_strlen() – returneaza lungimea unui sir
- mb_substr() – obtine un substring al unui string dat
- mb_strpos() – obtine pozitia unui subsir intr-un sir dat

Functiile UNICODE sint folosite pentru caractere slave, grecesti, arabe, chinezesti, japoneze, coreene si multe altele.

Despre lucrul cu stringuri se mai pot spune multe alte lucruri, insa o sa ma restring la citeva din lucrurile de baza. Nu uitati sa consultati documentatia limbajului pentru toti parametrii posibili ai acestor functii, eventual cu alte exemple de lucru cu aceste functii.

9. Siruri

Un sir (array) este o colectie de valori avind acelasi tip, accesate cu ajutorul unei variabile si a unui index. Exista mai multe feluri de a defini un sir. Cel mai frecvent este creat folosind array():

```
$a=array(1,2,3);
```

Asta va crea un sir cu 3 elemente cu indecsi de la 0 la 2. Accesul unui element se face asa: $a[0] sau $a[1].

Sau se pot crea siruri cu anumiti indecsi:

```
$a = array(3=>1, 5=>2);
```

Aceasta va crea un sir cu 2 elemente avind doar indecsii 3 si 5. Un element se va apela ca $a[3] sau $[5]. Folosirea altui index fa genera erorare:

```
echo $a[3]." - ".$a[4];
```

Va afisa:

1 - PHP Notice: Undefined offset: 4

Se pot insa crea siruri cu indecsi ne-numerici (string):

```
$a = array(‘unu’ => 1, ‘doi’ => 2); // sau folosind “ nu ‘
```

Apoi se folosesc:

```
echo $a[‘unu’];
```

Se pot adauga alte elemete la un sir:

```
$a= array(“a”);
$a[]=”b”;
$a[]=”c”;
echo $a[2]; // va afisa “c”
```

Elementul nou adaugat va avea ultimul index folosit +1.

Pot exista si combinatii:

```
$a = array("foo" => "bar", 10 => true);
$a[]="nou";
echo "{$a['foo']} - {$a[10]} - {$a[11]} <br>\n";
print_r($a);
```

Va afisa:

bar - 1 - nou
Array ([foo] => bar [10] => 1 [11] => nou)

Se pot vedea aici mai multe lucruri:

- indecsii pot fi atit string cit si numerici;
- noul element adaugat va avea indexul ultimul index numeric +1, daca exista asa ceva, altfel va fi 0 (in cazul nostru va fi 11, pentru ca ultimul index numeric este 10);
- se pot afisa elementele unui sir ca variabile intr-un string, inclusiv caractere de control (‘\n’) sau cod HTML (“
”);
- se poate folosi print_r() pentru afisarea intregului sir cu indecsi cu tot, de obicei pentru debug.

Tot asa cum se pot adauga elemente la un sir se pot si sterge folosind unset():

```
unset($a[11]);
```

Se pot crea *siruri multidimensiunale*. De exemplu:

```
$a=array(array(1,2,3),array(4,5,6));
```

Asta va crea matricea:
(1,2,3)
(4,5,6)

Un element poate fi accesat ca $a[1][2] ceea ce va returna 6.
Sau pot fi si combinatii ca:

```
$a = array(“sir” => array(1,2,3));
```

Apoi se poate folosi ca $a[“sir”][2] ce va returna 3.

Numarul elementelor unui sir poate fi obtinut cu functia count().
Pentru exemplul de mai sus:

```
echo count($a);
// afiseaza 1 pentru ca este un singur element

echo count($a[“sir”]); // afiseaza 3
```

Pentru a obtine toate elementele unui sir se poate folosi un ciclu for():

```
for($i=0;$i<count($a[“sir”]);$i++)
  echo $a[“sir”][$i].”<br>\n”;
```

Dar exista si o alta solutie mai eleganta cu foreach():

```
foreach($a[“sir”] as $key => $val) {
  echo “a[\”sir\”][{$key}]={$val}”;
  echo “<br>\n”;}
```

Va afisa:
a["sir"][0]=1
a["sir"][1]=2
a["sir"][2]=3

In codul de mai sus $key si $val pot fi folosite cu orice nume de variabile (nu neaparat acestea).

Sa reluam aici un exemplu dat la foreach():

```
$a=array(1,2,3);
foreach($a as $val) echo $val.” “;
foreach($a as $cheie => $val) {
  echo “\n<br>cheia: “.$cheie.” – valoare: ”.$val”; }
```

Exista o varianta alternativa la aceasta constructie care va fi explicata la capitolul despre siruri.

Acelasi lucru se poate rescrie asa:

```
$a = array("one", "two", "three");
reset ($a);
while (list($cheie, $val) = each ($a)) {
    echo “\n<br>cheia: “.$cheie.” – valoare: ”.$val”;
}
```

Aici sint folosite citeva functii:

- reset() – initializeaza indexul unui sir la 0

- each() – returneaza cheia si valoarea unui element si trece la urmatorul

Sa vedem aici citeva utilizati a operatorilor si functiilor pentru siruri:

```
<?php
$a = array("a" => "apple", "b" => "banana");
$b = array("a" => "pear", "b" => "strawberry", "c" => "cherry");

$c = $a + $b; // Reuniunea $a si $b
echo "<br>Reuniunea \$a si \$b: \n";
print_r($c);

$c = $b + $a; // Reuniunea $b si $a
echo "<br>Reuniunea \$b si \$a: \n";
print_r($c);
?>
```

Asta o sa afiseze

Reuniunea $a si $b: Array ([a] => apple [b] => banana [c] => cherry)
Reuniunea $b si $a: Array ([a] => pear [b] => strawberry [c] => cherry)

Exista apoi destul de multe functii referitoare la siruri (tablouri) multe care incep cu ‘array_’ dar nu numai, si anume pentru intersectie, diferenta, reuniune, extrage un element aleator, sortare, inversare, rotire, etc. O sa dau mai jos citeva din ele:

- array_change_key_case -- Returnează un tablou cu toţi indecşii literali reprezentaţi ca minuscule sau majuscule
- array_chunk -- Împarte un tablou în mai multe tablouri
- array_combine -- Creaza un sir in alte 2 siruri: unul pentru cheii si altul pentru valori
- array_count_values -- Numără valorile diferite unui tablou
- array_diff -- Calculează diferenţa tablourilor
- array_fill -- Umple un tablou cu valori
- array_filter -- Filtrează elementele unui tablou utilizând o funcţie callback
- array_flip – Schimba intr-un tablou cheile cu valorile si invers
- array_intersect – Calculeaza intersectia tablourilor
- array_key_exists – Verifica existenta unei cheii date
- array_keys – Returneaza un sir cu toate cheiile tabloului

- array_map -- Aplica o functie data tuturor elementelor unui tablou
- array_merge_recursive -- Uneste recursiv 2 sau mai multe tablouri
- array_merge – Uneste 2 sau mai multe tablouri
- array_multisort – Sorteaza tablouri multiple sau tablouri multi-dimensionale (matrici)
- array_pad -- Pad array to the specified length with a value
- array_pop – Extrage ultimul element al tabloului si scurteaza tabloul cu un element
- array_push -- Adauga un nou element la sfirsitul tabloului
- array_rand -- Selecteaza unul sau mai multe elemente aleator din tablou
- array_reduce -- Reduce un tablou folosind o functie data
- array_reverse -- Inverseaza elementele sirului
- array_search -- Cauta un element in tablou
- array_shift -- Roteste elementele unui tablou
- array_slice – Extrage o parte dintr-un tablou
- array_splice -- Elimina o parte dintr-un tablou si o inlocuieste cu altceva
- array_sum -- Insumeaza elementele unui tablou
- array_unique – Elimina elementele duplicate din tablou
- array_values – Returneaza un tablou cu toate valorile tabloului
- array_walk -- Aplica o functie data tuturor elementelor tabloului
- array -- Creează un tablou
- count – Numara elementele unui tablou
- current – Returneaza elementul curent in tablou
- each -- Returneaza elementul curent ca o combinatie de cheie + valoare si avanseaza la urmatorul element
- end -- Muta elementul curent la sfirsitul tabloului
- in_array -- Returneaza TRUE daca valoarea data este in tablou
- next -- Avanseaza la urmatorul element in tablou
- pos – Obtine pozitia curenta intr-un tablou
- prev – Merge la elementul anterior in tablou
- range -- Creaza un tablou dintr-o parte a altui tablou
- reset -- Merge la primul element din tablou
- rsort – Sorteaza un tablou in ordine inversa
- shuffle – Aranjeaza aleator elementele unui tablou
- sizeof – Obtine numarul elementelor
- sort – Sorteaza un tablou

Lista completa a tuturor functiilor o gasiti la http://php.net/array cu alte exemple si comentarii.

O sa dau un scurt exemplu de utilizare a citorva din aceste functii:

```
<?php
$a=array(10,20,30);
$b=array(10 => "a",20 => "b", 30=>30);

echo "\n<br>Diferenta: ";
print_r(array_diff($a,$b));

echo "\n<br>Intersectia: ";
print_r(array_intersect($a,$b));

echo "\n<br>Unirea: ";
print_r(array_merge($a,$b));

echo "\n<br>Numarul elem. tablou a: ";
print(count($a));

echo "\n<br>Numarul elem. diferite pt. a unit cu b: ";
print_r(array_count_values(array_merge($a,$b)));

echo "\n<br>Valorile tablou b: ";
print_r(array_values($b));

echo "\n<br>Cheile tablou b: ";
print_r(array_keys($b));

echo "\n<br>Suma elem. tablou a: ";
print(array_sum($a));

echo "\n<br>Tablou a inversat: ";
print_r(array_reverse($a));

echo "\n<br>Copie a tabloului a sortata invers: ";
$a2=$a;
rsort($a2);
print_r($a2);
unset($a2);
?>
```

Acestea vor afisa:

Diferenta: Array ([0] => 10 [1] => 20)
Intersectia: Array ([2] => 30)
Unirea: Array ([0] => 10 [1] => 20 [2] => 30 [3] => a [4] => b [5] => 30)

Numarul elem. tablou a: 3
Numarul elem. diferite pt. a unit cu b: Array ([10] => 1 [20] => 1 [30] => 2 [a] => 1 [b] => 1)
Valorile tablou b: Array ([0] => a [1] => b [2] => 30)
Cheile tablou b: Array ([0] => 10 [1] => 20 [2] => 30)
Suma elem. tablou a: 60
Tablou a inversat: Array ([0] => 30 [1] => 20 [2] => 10)
Copie a tabloului a sortata invers: Array ([0] => 30 [1] => 20 [2] => 10)

Se poate vedea sursa HTML rezultatelor afisate cu toate elementele aranjate altfel (plain-text, nu HTML).

Dupa cum vedeti lucrul cu siruri (tablouri) in PHP nu este complicat, fiind mult ajutat de existenta unor functii utile pentru diverse operatiuni. Trebuie doar sa vedeti daca nu cumva ceea ce doriti sa faceti are sau nu o functie gata implementata, care va scuteste de multa munca.

10. Functii

Urmatorul pas in lucrul cu PHP sint functiile. Pina acum am dat exemplu de multe functii implementate de limbaj. Insa deseori sint folosite functii create de programator care primesc sau nu diversi parametri, fac niste operatiuni si returneaza sau nu un rezultat.

O functie are un nume, poate avea sau nu parametri si poate intoarce sau nu un rezultat. Poate arata asa:

```
function foo ($arg_1, $arg_2, /* ..., */ $arg_n)
{
    echo "Exemplu de functie.\n";
    return $retval;
}
```

Unde:
- foo este numele functiei;
- $arg_1, $arg_2, …, $arg_n sint parametri;
- $retval este valoarea returnata

Apoi se poate folosi ca foo(1,2,3);

Daca functie pur si simplu nu are parametri se scrie
function foo() { … }

Daca nu returneaza nimic se sterge linia cu ‘return’.

Asa ca in final functie minimala poate fi:

```
function foo ()
{
    echo "Exemplu de functie.\n";
}
```

La apelul ei cu foo() va afisa stringul din echo si se va termina.

Sa luam un alt exemplu:

```
function foo($n) {
  echo $n;
  $n=10;
}
```

```
$i=5;
foo($i);
echo " - ".$i;
```

Aceasta va afisa: 5 - 5

Pasii care se executa sint urmatorii: se declara variabila $i initializata cu 5 care este folosita in apelul functiei. Parametrul $n va primi valoarea lui $i care este afisat, dupa care se modifica (=10). Apoi functia se termina si controlul e dat codului apelant unde e afisata valoarea veche a lui $i. Dupa cum se vede, valoarea schimbata in functie nu s-a transferat in afara functiei.

In acest caz se spune ca transmiterea parametrilor functiei s-a facut prin valoare, modul standard de transmitere in PHP. Exista insa si o alta solutie: transmiterea prin referinta cind valoarea schimbata in corpul functiei este transmis in codul apelant.

```
function foo(&$n) {
  echo $n;
  $n=10;
}

$i=5;
foo($i);
echo " - ".$i;
```

Va afisa: 5 – 10

Singura schimbare este “&” adaugat inaintea parametrului $n care specifica transferul prin referinta a parametrilor.

Un alt lucru posibil pentru parametrii functiei este initializarea lor la declararea functiei:

```
function foo($n=5) {
  echo $n;
}

foo();
```

Asta va afisa: 5
chiar daca functia este apelata fara parametri curenti.

Singura restrictie atunci cind exista parametri de functie initalizati este ca acestia trebuie sa fie grupati ca ultimii parametri dati, nu la inceputul listei de parametri.
De exemplu:

```
function foo($n=5, $m) { … }
```

nu este valid. Trebuie rescris ca:

```
function foo($m, $n=5) { … }
```

De exemplu:

```
function foo($m,$n=5) {
  echo $m+$n;
}

foo(3);
echo " - ";
foo(3,6);
```

Va afisa: 8 - 9

Prima data este initializat doar $m cu valoarea 3, iar $n va avea valoarea default (=5). Apoi se va apela cu valoarea 3 pentru $m si 6 pentru $n.

Valoarea returnata de o functie este data cu ‘return’:

```
function foo($n) { return $n+1; }
```

Apoi se poate folosi de exemplu:

```
echo foo(5);
$i=foo(10);
```

Care va afisa: 6
Si va initializa variabila $i cu 11

Este posibil ca o functie sa returneze o referinta la o variabila nu o valoare, insa vom lasa acest exemplu deoparte.

Alte lucruri ce se pot folosi in PHP sint nume de functii variabile.

```
<?php
function foo() { echo "In foo()<br />\n"; }
```

```
function bar($arg = '')
{ echo "In bar(); argument was '$arg'.<br />\n"; }

function echoit($string)
{ echo $string; }

$func = 'foo';
$func();        // cheama foo()

$func = 'bar';
$func('test');  // cheama bar() cu un parametru

$func = 'echoit';
$func('test');  // cheama echoit() cu un parametru
?>
```

Dupa cum vedeti $variabila() cheama functia cu numele tinut in variabila respectiva (trebuie doar sa puneti parantezele de functie la o variabila). Numele variabilei nu este important, doar valoarea ei sa contina numele functiei si sa nu uitati parantezele, eventual si parametrii (daca are).

Un alt lucru mai avansat in lucrurl cu functiiel este posibilitatea declararii functiilor fara parametrii si accesarii parametrilor folositi la apelul functiei cu ajutorul functiilor auxiliare func_num_args(),func_get_arg() si func_get_args():

```
<?php
function test() {
echo "Parametrii dati sint:<br>\n";
for($i=0;$i<func_num_args();$i++)
  echo "param[".$i."]=".func_get_arg($i)."<br>\n";
}

test(3,2,1);
?>
```

Va afisa:

Parametrii dati sint:
param[0]=3
param[1]=2
param[2]=1

11. Lucrul cu forme HTML

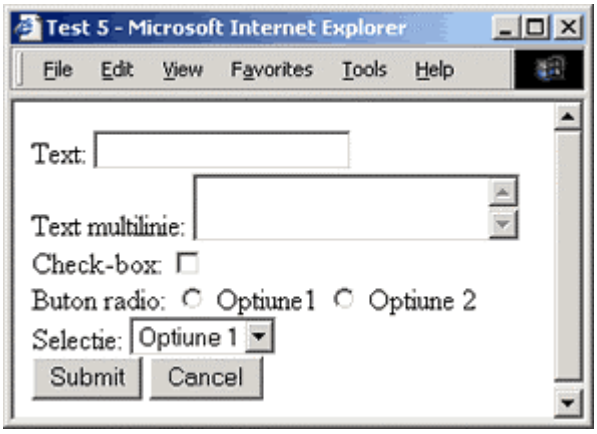
PHP se dovedeste foarte util cind se pune problema prelucrarii datelor pasate din forme HTML. Nu o sa prezentam aici toate detalii despre controalele disponibile in formele HTML, dar o sa incep cu o scura prezentare a celor mai folosite:

- text-box: <input type=text>
- text-box multilinie (textarea): <textarea></textarea>
- check-box: <input type=checkbox>
- buton radio: <input type=radio>
- butoane: <input type=button> sau <input type=submit> sau <input type=cancel>
- cimp ascuns: <input type=hidden>
- lista de selectie: <select></select>
- upload de fisiere: <input type=file>

Sa consideram urmatorul fisier de test (test5.php):

```
<html>
<head><title>Test 5</title></head>
<body>
<form name="form1" method="post" action="test5.php">
  <p>Text: <input type="text" name="textfield"><br>
    Text multilinie:
     <textarea name="textarea"></textarea><br>
    Check-box: <input type="checkbox" name="checkbox"
value="checkbox"><br>
    Buton radio:
    <input type="radio" name="radiobutton"
value="optiune1">Optiune1
    <input type="radio" name="radiobutton"
value="optiune2">Optiune 2 <br>
    Selectie: <select name="select">
      <option value="1">Optiune 1</option>
      <option value="2">Optiune 2</option>
    </select><br>
    <input type="submit" name="Submit" value="Submit">
    <input name="Cancel" type="reset" value="Cancel">
  </p>
</form>
</body>
</html>
```

Pagina va arata asa:



Puteti introduce valori in cimpurile date, selecta optiuni si apoi apasa butonul ‘Submit’ insa veti vedea ca nu se intimpla nimic cu valorile selectate. Sau daca apasati butonul ‘Cancel’ valorile sint sterse imediat, fara alte actiuni.

Nu se intimpla nimic pentru ca este necesar cod PHP pentru prelucrarea valorilor transmise din forma. Asa cum am spus aceasta prelucrare nu se poate face folosind HTML simplu ci doar cod executat pe server, gen PHP, ASP, Perl sau altele.

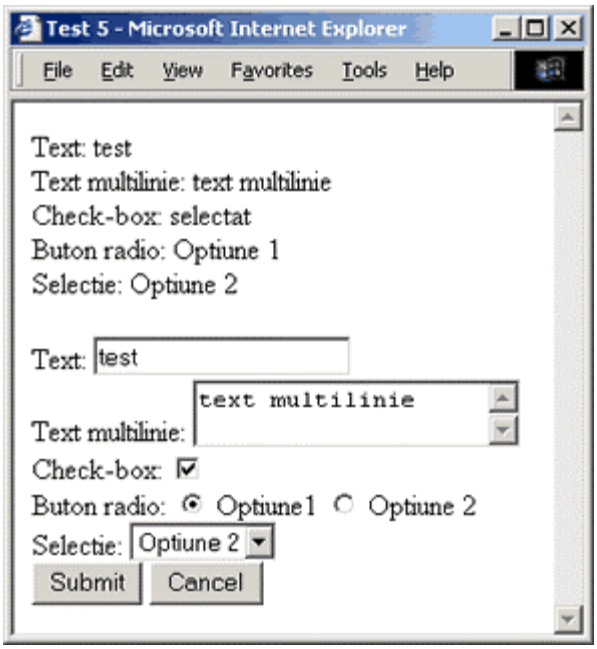
Carei pagini ii sint pasate valorile de prelucrat, introduse in controalele formei? Este vorba de pagina specificata in proprietatea “action” a formei:
<form … action=”test5.php”>
Se poate pasa valorile spre prelucrare aceleiasi pagini sau alteia (nu uitati ca ati numit pagina test5.php).

Sa aratam acum noua pagina, cu cod PHP de prelucrarea parametrilor:

```
<html>
<head><title>Test 5</title></head>
<body>
<?php
if(isset($_POST['Submit'])) {
   echo "Text: {$_POST['textfield']}<br>\n".
"Text multilinie: {$_POST['textarea']}<br>\n".
     "Check-box: ".
($_POST['checkbox']=="checkbox"?"selectat":"neselectat")."<b
r>\n".
     "Buton radio: ".
($_POST['radiobutton']=="optiune1"?"Optiune 1":"Optiune
2")."<br>\n".
```

```
     "Selectie: ".
($_POST['select']=="1"?"Optiune 1":"Optiune 2")."<br>\n";
}
?>
<form name="form1" method="post" action="test5.php">
  <p>Text: <input type="text" name="textfield"
value="<?php echo $_POST['textfield']; ?>"> <br>
    Text multilinie:
    <textarea name="textarea"><?php echo $_POST['textarea'];
?></textarea>
   <br>Check-box:
    <input type="checkbox" name="checkbox" value="checkbox"
    <?php if($_POST['checkbox']=="checkbox") echo "checked";
?>>
    <br>Buton radio:
    <input type="radio" name="radiobutton" value="optiune1"
<?php if($_POST['radiobutton']=="optiune1") echo "checked";
?>>
    Optiune1
    <input type="radio" name="radiobutton" value="optiune2"
<?php if($_POST['radiobutton']=="optiune2") echo "checked";
?>>
    Optiune 2 <br>
    Selectie: <select name="select">
      <option value="1" <?php if($_POST['select']=="1") echo
"selected"; ?>>Optiune 1</option>
      <option value="2" <?php if($_POST['select']=="2") echo
"selected"; ?>>Optiune 2</option>
    </select><br>
    <input type="submit" name="Submit" value="Submit">
    <input name="Cancel" type="reset" id="Cancel"
value="Cancel">
  </p>
</form>
</body>
</html>
```

Pagina ar putea fi:



Sint mai multe lucruri de urmarit aici:

- intii este verificat daca pagina este reincarcata dupa postarea inapoi a parametrilor verificind valoarea butonului 'Submit' cu isset(). La prima incarcare acesta e stringul null.
- daca s-au pasat parametrii inapoi se face afisarea valorilor selectate
- apoi se face afisarea din nou a formei, de data aceasta afisind valorile selectate dinainte. De regula, valorile pasate spre prelucrare inapoi in pagina nu mai sint afisate automat. In acest caz am folosit de exemplu:
  <input type="text" name="textfield"
  value="<?php echo $_POST['textfield']; ?>">

Dupa cum se poate vedea sint importante numele controalelor din forma care sint folosite apoi la apelul valorilor pasate inapoi spre prelucrare, folosind $_POST: $_POST['textfield'], $_POST['checkbox'], $_POST['select'].

Acestea se putea denumi si altfel (la fel si pentru celelalte controale):

```
<input type=text name=txtNume
value="<?php echo $_POST['txtNume']; ?>">
```

De asemenea este importanta si valoarea pasata inapoi paginii pentru diverse controale. La controale text valoarea pasata este textul introdus in controlul respectiv, la liste de selectie esta valoarea optiunii

selectate, la checkbox este valoarea data in ‘value’ (<input type=checkbox name=chkTest value=”1”>), sau daca aceasta lipseste va fi “on” daca e selectat si stringul null daca nu e selectat (<input type=checkbox name=chkTest2> se va putea testa if($_POST[‘chkTest2’]==”on”) …).

La butoane radio valoarea este cea a optiunii selectate (<input type=radio name=rdTest value=”1”>). Un grup de butoane radio care va schimba valoarea intre ele la selectie au toate acelasi nume. Pentru alt grup separat se foloseste nume diferit:

```
<input type=radio name=grup1 value=”1”>Optiune 1
<input type=radio name=grup1 value=”2”>Optiune 2<br>
<input type=radio name=grup2 value=”B”>Optiune A
<input type=radio name=grup2 value=”B”>Optiune B<br>
```

Valorile selectate se pot testa cu $_POST[‘grup1’], respectiv $_POST[‘grup2’] care au valorile null (“”) daca nimic nu e selectat dintr-un grup, “1” / “2”, respectiv “A” / “B”.

Un capitol aparte in lucrul cu forme o constituie ***upload-ul de fisiere***. Pentru asta exista un control special HTML: <input type=file> Acesta o sa afiseze automat un text-box si un buton pe care scrie ‘Browse’ (neschimbabil) pentru selectia fisierului de uploadat de de calculatorul dvs. Din motive de securitate nu se poate face nimic automat si nu se poate initializa controlul. Insa pentru aceste cazuri si forma este una speciala avind nevoie de un parametru: enctype="multipart/form-data"

Prelucrarea fisierului uploadat pe server se face cu variabila $_FILES care are citeva valori predefinite:

$_FILES['file']['name']
: Numele fisierului original

$_FILES['file']['type']
: Mime type pentru fisierul de uploadat. Exemplu "image/gif", “image/pjpeg”.

$_FILES['file']['size']
: Marimea fisierului in bytes.

$_FILES['userfile']['tmp_name']
: Numele temporar sub care e copiat pe server

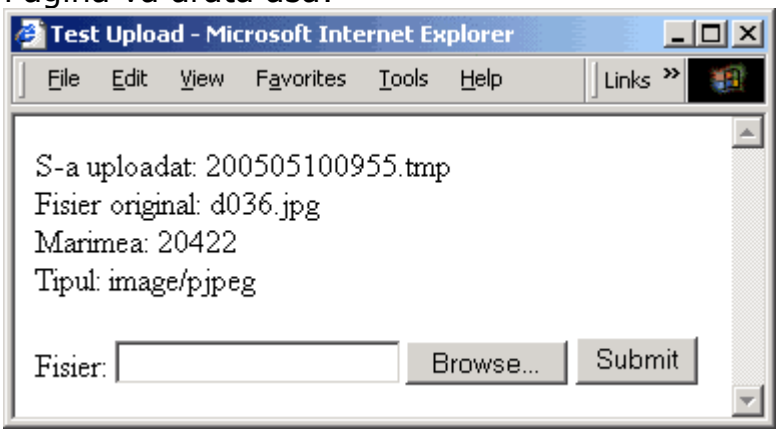
$_FILES['file']['error']
: Codul erorii asociat uploadul, daca sint erori

In acest caz s-a presupus ca avem un control file numit ‘file’. Daca aveam
<input type=file name=poza> se folosea $_FILES['poza']['name'], s.a.m.d.

Sa vedem acum un exemplu complet (fisierul numit testupload.php):

```
<html>
<head><title>Test Upload</title></head>
<body>
<?php
if(isset($_POST['Submit'])) {
if(($_FILES['file']['size']<100000)&&($_FILES['file']['size'
]>0)) {
  $filename=date("Ymdhi").".tmp";
  move_uploaded_file($_FILES['file']['tmp_name'],
$filename);
  echo "<p>S-a uploadat: {$filename}
     <br>Fisier original: {$_FILES['file']['name']}
     <br>Marimea: {$_FILES['file']['size']}
     <br>Tipul: {$_FILES['file']['type']}</p>\n";
  }
}
?>
<form name="form1" method="post" action="testupload.php"
enctype="multipart/form-data">
  <p>Fisier:
    <input type="file" name="file">
    <input type="submit" name="Submit" value="Submit">
  </p>
</form>
</body>
</html>
```

Pagina va arata asa:



Dupa cum vedeti puteti testa lungimea fisierelor sau puteti schimba numele fisierelor (aici este generat automat din data curenta).

Aveti insa grija cum folositi aceste optiuni pentru a nu deschide brese de securitate pe site-ul dvs. Este posibil ca cineva sa incerce sa uploadeze fisiere PHP pe care apoi sa le execute, sau alte tipuri de fisiere (HTML, JavaScript, etc.) Asa ca intotdeauna sa verificati tipul fisierelor si marimea lor pentru a nu avea surprize.

12. Lucrul cu fisiere

Din PHP se pot deschide si apoi scrie sau citi fisiere, de regula aflate pe server dar si de pe alte calculatoare sau site-uri. Pentru aceste operatiuni sint folosite citeva functii: fopen(), fclose(), fwrite(), fread(), fgets(), file(), flush(), filesize() si altele.

Exemplu de deschidere a unor fisiere in diverse moduri:

```
<?php
$handle = fopen("/home/rasmus/file.txt", "r");
$handle = fopen("/home/rasmus/file.gif", "wb");
$handle = fopen("http://www.example.com/", "r");
$handle =
fopen("ftp://user:password@example.com/somefile.txt", "w");
?>
```

Primul parametru este fisierul de deschis, iar al doilea parametru reprezinta modul de deschidere a fisierului. Acestea pot fi:

“r” – fisier deschis doar pentru citire
“r+” – fisier deschis pentru scris si citit
“w” – fisier deschis doar pentru scriere
“w+” – fisier deschis pentru scris si citit, daca nu exista fisierul o sa fie creat
“a” – fisierul deschis pentru adaugare la sfirsit
“a+” - fisierul deschis pentru adaugare la sfirsit, daca nu exista fisierul o sa fie creat
“t” – fisier deschis in mod text
“b” – fisier deschis in mod binar

De asemeena, acestea se pot combina (exemplu “rb”, “wb”).

Cu acestea se obtine un id ($handle) care apoi se foloseste in celelate functii:

```
$filename = "/usr/local/something.txt";
$handle = fopen($filename, "r");
$contents = fread($handle, filesize($filename));
fclose($handle);
```

Pentru Windows, pentru a deschide un fisier binar trebuie specificat parametrul “b”, iar backslash-ul trebuie dublat:

```
$filename = "c:\\files\\somepic.gif";
$handle = fopen($filename, "rb");
$contents = fread($handle, filesize($filename));
fclose($handle);
```

Acelasi lucru se poate realiza mai simplu folosind file(), doar ca asta returneaza un sir de stringuri pentru fiecare linie, nu un simplu string:

```
$contents= file('http://www.example.com/');
$contents= file(‘/usr/local/something.txt');
```

Scrierea intr-un fisier se face la fel de simplu:

```
<?php
$filename = 'test.txt';

if (is_writable($filename)) {
    if (!$handle = fopen($filename, 'a')) {
         echo "Nu pot deschide fisierul ($filename)";
         exit;
    }
    if (fwrite($handle, date("d/m/Y h:i")."\r\n") === FALSE)
{
        die("Nu poate scrie ($filename)");
    }
    fclose($handle);
    print_r(file($filename));
} else {
    echo "Fisierul $filename nu se poate scrie.";
}
?>
```

Aveti doar grija sa creati intii fisierul ‘test.txt’ in folderul curent (al paginii PHP).
Se pot urmari aici citeva lucruri:

- se poate testa existenta si posibilitatea scrierii unui fisier cu is_writable(). Pot fi probleme de acces a unor fisiere, caz in care trebuie schimbate drepturile de acces pe server (pe Linux/Unix cu CHMOD, vezi manualul);
- in mod normal trebuie verificate erorile returnate de deschiderea fisierului sau scriere lui;
- se poate afisa un mesaj si opri scriptul in 2 feluri: echo + exit sau die();
- fisierul trebuie inchis dupa folosire

Dupa incarcarea paginii de citeva ori verificati fisierul test.txt de pe server.

Toate aceste operatiuni sint separat de includerea unor fisiere PHP sau de alta natura in codul paginii (realizat cu include() sau require()). Multe din informatiile scrise sau citite din fisiere pot fi pastrate permanent in baze de date, insa despre acestea in capitolul urmator.

O alta categorie de functii se refera la manipularea fisierelor de pe server, nu citirea sau scrierea lor. Sint operatiuni de cautare a unor fisiere (opendir() – readdir()), verificare daca este fisier sau director (is_file() – is_dir()), existenta unui fisier (file_exists()) si altele.

De exemplu, pentru afisarea tuturor fisierelor si directoarelor din folderul curent:

```
<?php
$dir = "./";
if (is_dir($dir)) {
    if ($dh = opendir($dir)) {
        while (($file = readdir($dh)) !== false) {
            echo "fisier: $file : tip: " .
                  filetype($dir . $file) . "\n";
        }
        closedir($dh);
    }
}
?>
```

Exista multe alte operatiuni care se pot face cu fisiere, insa nu intram acum in toate detaliile. Ne restringem aici la lucrurile esentiale, pentru a nu ne pierde in detalii si complica prea mult explicatiile.

13. MySQL

In sfirsit am ajuns la capitolul cel mai interesant pentru multi: lucrul cu baze de date. Voi exemplifica aici doar lucru cu MySQL. Exista si alte posibilitati, cel mai frecvent dupa MySQL fiind PostgreSQL (asemenator), dar si altele.

Deseori, pentru pastrarea si apoi regasirea informatiilor se folosesc baze de date. MySQL este un server de baze de date simplu si eficient, care lucreaza foarte bine cu PHP. Foarte multe site-uri au in spate combinatia PHP + MySQL.

De cele mai multe ori MySQL este gestionat chiar din browser cu ajutorul phpMyAdmin un script PHP pentru administrarea MySQL. Dar exista si alte solutii sau chiar operatiuni din linia de comanda pe server. Se pot crea baze de date, care au tabele cu cimpuri, si eventual indecsi. Nu voi intra in detalii despre bazele de datele relationale as acum este MySQL, spun doar ca operatiunile permise se pot face folosind SQL.

SQL este un limbaj simplu de interogare, modificare sau stergere a datelor, dar si de manipulare a structurii bazei de date. Exemple simple:

SELECT * FROM Tabela
    Selecteaza tot continutul tabelei Tabela

INSERT INTO Tabela(cimp1,cimp2) VALUES(1,'string')
    Insereaza o noua inregistrare in tabela

UPDATE Tabela SET counter=counter+1 WHERE id=1
    Incrementeaza cimpul counter pentru id=1

DELETE FROM Tabela WHERE Nume LIKE 'A%'
    Sterge inregistrarile pentru care cimpul Nume incepe cu 'A'

Pentru crearea sau modificarea structurii datelor se poate scrie:

```
CREATE TABLE users (
   id int(11) NOT NULL auto_increment,
   Nume varchar(30) default NULL,
   DataInregistrarii date,
   PRIMARY KEY  (id)
)
```

ALTER TABLE users ADD Email varchar(50);
Adauga un cimp nou la o tabela existenta

ALTER TABLE users DROP DataInregistrarii;
Sterge un cimp existent

CREATE INDEX IDX_Nume ON users(Nume);
Creaze un index.

DROP TABLE users;
Sterge tabela users

Pentru toate comenzile disponibile in MySQL vedeti manualul acestuia sau documentatia online de la http://dev.mysql.com/doc/mysql/en/index.html . De asemenea nu voi intra in detalii despre toate tipurile de date disponibile in MySQL, desi cele mai frecvent folosite sint:

- int - valoare intreaga (32 biti)
- tinyint – valoare intreaga mica (16 biti)
- varchar() – string cu lungime maxima data
- text – string de orice lungime, de obicei pt. texte mari
- float – valoare cu zecimale
- date – data calendaristica

Sau asa cum am spus, cel putin pentru modificarile structurii bazelor de date se MySQL poate folosi phpMyAdmin (de la http://www.phpMyAdmin.net/) care mai mult ca sigur il aveti instalat pe hostingul site-ului dvs.

Insa toate aceste se pot executa si cu ajutorul comenzilor PHP, care are functii speciale pentru lucrul cu MySQL. Pasii care se executa deseori sint: conectarea la baza de date folosind un user name si parola, deschiderea unei baze de date (pot fi mai multe), interogarea SQL, eliberarea elementelor extrase din tabele, inchiderea conexiunii:

```
<?php
/* Conectarea si selectarea bazei de date  */
$link =
 mysql_connect("mysql_host", "mysql_user", "mysql_password")
 or die("Conectarea a esuat : " . mysql_error());
echo "Conectare cu succes";
mysql_select_db("my_database")
      or die("Nu se poate selecta baza de date");
```

```
/* Executa comanda SQL */
$query = "SELECT * FROM my_table";
$result = mysql_query($query)
       or die("Interogare esuata : " . mysql_error());

echo “Au fost returnate “.
      mysql_num_rows($result).” inregistrari\n”;

/* Executa prelucrarea – aici afiseaza ca o tabela HTML */
echo "<table>\n";
while ($line = mysql_fetch_assoc($result)) {
    echo "\t<tr>\n";
    foreach ($line as $col_value) {
        echo "\t\t<td>$col_value</td>\n";
    }
    echo "\t</tr>\n";
}
echo "</table>\n";

/* Elibereaza inregistrarile */
mysql_free_result($result);

/* Inchide conexiunea */
mysql_close($link);
?>
```

Pentru acest caz se inlocuieste parametrii care reprezinta numele serverului (de obicei “localhost”), numele si parola de acces, iar apoi numele bazei de date. Daca se returneaza erorare la una din functiile de mai sus se afiseaza eroarea si se opreste executia cu die(). Eroarea se obtine cu mysql_error().

Dupa cum se poate vedea functiile folosite sint:

- mysql_connect() – conectarea la baza de date
- mysql_select_db() – selecteaza o baza de date din mai multe posibile
- mysql_query() – executa o comanda SQL
- mysql_num_rows() – obtine numarul inregistrarilor returnate
- mysql_fetch_assoc() sau mysql_fetch_array() – returneaza o linie ca sir asociativ (cu cheii de tip string) sau sir normal (indici numerici)
- mysql_free_result() – elibereaza inregistrarile returnate
- mysql_close() – inchide conexiunea

Pentru a obtine valoarea unui cimp doar se poate scrie:

```
$line = mysql_fetch_assoc($result);
echo $line[‘nume_cimp1’];
```

sau

```
$line = mysql_fetch_array($result);
echo $line[0];
```

Exista doar un lucru de mentionat aici: se poate lucra cu conexiuni permanente folosind mysql_pconnect(). Aceasta e asemanatoare cu mysql_connect doar ca intii se verifica daca nu cumva exista o conexiune deja deschisa nefolosita, iar la final conxiunea nu este inchisa ci este pastrata pentru alte posibile conexiuni ulterioare. Atentia doar la faptul ca numarul conexiunilor este totusi limitat si asta poate ingreuna accesul la baza de date. In special pentru site-urile cu trafic mai mare e bine de facut teste care din acestea lucreaza mai bine.

Alte functii utile care pot fi folosite destul de des:

- mysql_result() – returneaza valoarea unui cimp pe baza unei linii sau linii + coloane
- mysql_fetch_object() – returneaza o linie ca obiect. Exemplu de folosire:

```
$row = mysql_fetch_object($result);
echo $row->nume_cimp;
```

- mysql_fetch_row() – obtine un o linie ca un sir
- mysql_unbuffered_query() - executa mai rapid comenzi SQL atunci cind acestea nu returneaza inregistrari (UPDATE, DELETE, INSERT INTO, etc.)

Un exemplu de enumerare a tuturor bazelor de date si tabelelor:

```
<?php
if(!$link=mysql_connect('localhost', 'root', '')) {
    die('Num se poate conecta'); }

$db_list = mysql_list_dbs($link);
while ($row = mysql_fetch_object($db_list)) {
  echo "Baza de date: ".$row->Database . "\n";
  $result = mysql_list_tables($row->Database);
  while ($row_table = mysql_fetch_row($result)) {
```

```
    echo " - Tabela: $row_table[0]\n";
  }
  mysql_free_result($result);
}
?>
```

Exista 2 loop-uri: una pentru toate bazele de date si alta pentru toate tabelele unei baze de date. Am folosit 2 metode diferite de obtinere a unei linii din cele returnate: mysql_fetch_row() si apoi mysql_fetch_object(). Se putea folosi doar una, sau alta sau chiar mysql_fetch_assoc() sau mysql_fetch_array(). In acest caz se conecteaza local cu userul default fara parola.

Alte pagini mai complicate cu PHP si MySQL sint o combinatie a tuturor acestor lucruri, eventual chiar parametri pasati din forme HTML, lucru cu fisiere, includeri de fisiere cu parametri globali (user name, parola, etc.) sau multe alte lucruri.

Lista completa a tuturor functiilor PHP pentru lucrurl cu MySQL o gasiti la http://www.php.net/manual/en/ref.mysql.php , insa de cele mai multe ori procesul de mai sus se repeta in cele mai multe cazuri, cu mici variatiuni.

14. Lucruri mai avansate cu PHP

A. Sesiuni

Dupa cum am spus exista variabila predefinita $_SESSION care se ocupa de sesiuni. Insa ce sint sesiunile?

De obicei paginile web sint ‘stateless’, adica fiecare cerere este tratata ca o cerere noua fara sa se stie de cererile anterioare. In acest fel ar fi greu de accesat un site dupa o identificare prealabila cu un nume de utilizator si parola pentru ca valorile introduse s-ar pierde. Pentru evitarea acestor probleme serverele web au modalitati de a tine minte vizitele ulterioare ale altor pagini, totul fiind tratat ca o singura sesiune de lucru. Aceste sesiuni pot fi identificate (sa ai acces abia dupa logare) si trebuie sa poata mentine diverse variabile pe toata durata lor (exemplu: nume de utilizator, un id, parole ,etc.)

Aceste valori sint tinute $_SESSION si sint pastrate pe toata durata vizitei unui site, chiar daca se incarca pagini diferite. Modalitatea tehnica de mentinerea sesiunilor se face prin cookie sau parametri ascunsi sau vizibili in adresa paginii cu un id unic, pasat inainte si inapoi la fiecare cerere .Atit timp cit id-ul e acelasi e vorba de aceeasi sesiune.

Acest identificator este pastrat pe server si expira dupa un anumit timp. Este generat aleator astfel incit s anu poata fi ghicit si se foloseste si al alte limbaje server-side (ASP), nu doar PHP.

O sesiune trebuie deschisa, se poate inchide si exista alte functii de manipulare a lor:

- session_start() – initializeaza o sesiune
- session_register() – inregistreaza un nume de variabila ca facind parte din sesiune
- session_unregister() – dez-inregistreaza un nume de variabila dintr-o sesiune
- session_unset() – elibereaza toate datele tinute in sesiune

Un exemplu concret de folosire a sesiunilor si _$SESSION:

```
<?php
// testsession1.php
session_start();
echo 'Pagina #1';
```

```
$_SESSION['time'] = time();
echo '<br /><a href="testsession2.php?' . SID . '">pagina 2
cu sesiune</a>';
?>
```

Apoi urmatoarea pagina testsession2.php:

```
<?php
// testsession2.php
session_start();
echo 'Pagina #2<br />
     Tipul actual: '.date('Y m d H:i:s').
     '<br />Timpul initial (din sesiune): '.
     date('Y m d H:i:s', $_SESSION['time']);
?>
```

Puteti reincarca de citeva ori testsession2.php (F5) si o sa vedeti ca timpul initial ramine acelasi, fiind cel pastrat in variabila sesiune.

SID este un identificator de lucru pentru sesiuni care adauga automat identificatorul sesiunii daca este necesar. In acest caz se putea renunta la el. PHP determina in mod inteligent cum se face pastrarea sesiunii cu identificatorul unic de pasat in fiecare pagina. De cele mai mutle ori nu trebuie facut nimic pentru ca o sa fie administrat automat.

Un lucru de recomandat pentru sesiuni este sa tineti cit mai putine informatii in sesiune. De multe ori este destul un identificator. Toate informatiile sint pastrate pe server iar pentru un site cu mii de vizitatori poate ocupa destula memorie.

De multe ori o sa vedeti link-uri gen:
pagina.php?PHPSESSID=46207c992915148c965f877f430336f5

In aceste cazuri PHPSESID nu este altceva decit identificatorul sesiunii. Este posibil ca numele parametrului sa fie schimbat.

B. Redirectari

De multe ori este necesar redirectarea de la o pagina la alta, in functie de unele conditii. Pentru diverse exista diverse solutii: se poate face redirectarea direct de pe server sau folosind browserul client.

Cea mai directa cale este folosind functia header(). Asta trimite parametri in antetul paginii HTML trimisa in browser cu sepcificarea incarcarii altei pagini:

```
header(“Location: pagina_noua.php”);
header(“Location: http://alt_server.com/pagina_noua.php”);
```

Exista situatii cind veti obtine erorare la acesta redirectare pentru ca este permisa doar in partea de inceput a paginii, inainte sa fie trimis corpul raspunsului care browser (doar in anteteul raspunsului). Retineti s-o folositi la inceputul paginii, nu in mijlocul ei.

O alta metoda de redirectate poate fi facuta folosing browserul client:

```
echo ‘<META HTTP-EQUIV=Refresh CONTENT="0;
URL=http://www.server.com/pagina.php">’
```

Aceasta va reincarca pagina data dupa numarul de secunde dat (0 in acest caz), automat, fara interventia userului.

Sau se poate folosi JavaScript:

```
echo “<script language=JavaScript>
document.location.href=’pagina_noua.php’;
</script>\n”;
```

Metoda folosing header() este utila cind totul se vrea sa se faca transparent pentru vizitator, fara ca acesta sa-si dea seama ca a fost redirectata catre alta pagina.

Metodele cu META Refresh sau JavaScript pot fi utile cind se doreste afisarea unui mesaj scurt de redirectare, gen ‘asteptati putin pina se reincarca pagina’. Incarcarea noii pagini poate dura citeva secunde timp cit va fi vizibil textul initial.

C. Sockets

Exista situatii cind este nevoie de un mai mare control de conectare un unor servere folosind diverse protocoale. De exemplu chiar la alte servere web dar nu numai. Pentru asta exista lucrul cu sockets folosind functia fsocketopen():

```
<?php
$fp = fsockopen("www.example.com", 80, $errno, $errstr, 30);
if (!$fp) {
    echo "$errstr ($errno)<br />\n";
} else {
    $out = "GET / HTTP/1.1\r\n";
    $out .= "Host: www.example.com\r\n";
    $out .= "Connection: Close\r\n\r\n";

    fwrite($fp, $out);
    while (!feof($fp)) {
        echo fgets($fp, 128);
    }
    fclose($fp);
}
?>
```

Aici intervin mai multe probleme. Conexiunea se poate face practic pe orice calculator si pe orice port, atit timp cit conexiunile sint acceptate. Apoi insa trebuie folosit in mod direct comenzile protocolului respectiv. In exemplul dat e vorba de HTTP cu comanda GET, unde este ceruta o pagina si afisat raspunsul. Trebuie vazut in mod concret detalii privind protocolul folosit, ce comenzi are si ce parametri. Se poate folosi FTP, SMTP (emailuri), NNTP (servere de news), WHOIS, dar si alte protocoale. Pentru unele din acestea sint alte functii specializate, insa totul poate fi realizat si cu aceasta functie la nivel de jos.

Folosind aceasta functie puteti de exemplu POSTa diverse informatii al alte pagini si obtine raspunsul transmis de la acestea. Sau chiar face cereri multiple la un site care necesita login pentru a obtine diverse informatii, mentinind sesiunea de la o cerere la alta. Insa nu voi intra in alte detalii aici despre lucrul cu HTTP sau alte protocoale.

Anexa A: Legaturi utile

- http://php.net - PHP de download-at, documentatie, stiri, etc.
- http://rophp.net - comunitate romaneasca PHP
- http://phpromania.net - comunitate romaneasca PHP
- http://www.phpbb.ro - comunitatea romaneasca phpBB
- http://programare.org - comuitatea romaneasca despre programare, inclusiv PHP, MySQL dar si altele
- http://groups.yahoo.com/group/RoPHP/ - lista de discutii PHP romaneasca

Anexa B: Module si librarii

Diverse functii disponibile in PHP depind de prezenta sau nu a unor module. De exemplu functiile pentru MySQL depind de libraria MySQL care poate fi instalata sau nu. Informatii despre modulele sau librariile instalate se obtine cu functia phpinfo(). Cu aceasta functie o sa puteti vedea toti parametrii setati pentru module si daca un modul este prezent. Daca nu este listat aici inseamna ca nu este disponibil, iar functiile modulului sint de nefolosit.

Aceasta afiseaza un parametru numit: Configure Command care poate arata asa:

'./configure' '--with-apxs=/usr/local/apache/bin/apxs' '--with-xml' '--enable-bcmath' '--enable-calendar' '--with-curl' '--with-dom' '--with-dom-xslt' '--with-dom-exslt' '--enable-exif' '--enable-ftp' '--with-gd' '--with-jpeg-dir=/usr/local' '--with-png-dir=/usr' '--with-xpm-dir=/usr/X11R6' '--with-gettext' '--with-imap' '--with-imap-ssl' '--with-kerberos' '--enable-mbstring' '--enable-mbstr-enc-trans' '--enable-mbregex' '--with-mcrypt' '--with-mhash' '--with-ming=../ming-0.2a' '--enable-magic-quotes' '--with-mysql' '--with-openssl' '--enable-discard-path' '--with-pear' '--enable-xslt' '--with-xslt-sablot' '--enable-sockets' '--enable-track-vars' '--with-ttf' '--with-freetype-dir=/usr' '--enable-gd-native-ttf' '--enable-versioning' '--enable-wddx' '--with-xmlrpc' '--with-zlib'

Citeva module des folosite:

- bcmath – functii matematice cu mare precizie
- curl – functii de accesat site-uri externe prin HTTP, HTTPS, FTP, etc.
- domxml – functii pentru DOM XML
- gd – librarie grafica pentru modificat sau creat imagini, etc.
- mbstring – suport pentru caractere internationale (UNICODE)
- mcrypt – suport pentru functii de criptare / decriptare
- ming – librarie pentru generat fisiere Macromedia Flash (SWF)
- openssl – functii de lucru pentru site-uri securizate (SSL)
- pcre – expresii regulare compatibile Perl
- xml , xslt – suport pentru XML si XSL Transformations
- xmlrpc – suport pentru functii XML apelate de pe alte servere
- zlib – suport pentru comprezare zip a raspunsurilor

Evident exista si alte module, acestea find dintre cele mai uzuale.

Parametrii existenti se pot modifica in php.ini. Insa trebuie facut diferenta intre un parametru al unui modul existent sau lipsa modulului. De obicei daca va lipseste un astfel de modul intrebat hostingul.

Anexa C. Alte functii utile

- md5() – Codifica un string folosind algoritmul MD5 (pentru un text dat intotdeauna se obtine acelasi rezultat). Stringul nu mai poate fi decodificat pentru ca algorimul functioneaza doar pentru codificare nu si invers.
Exemplu:

```
echo md5(“test”);
```

Afiseaza: 098f6bcd4621d373cade4e832627b4f6

- nl2br() – inlocuieste caracterele de linie noua cu tagul HTML “
”. Este foarte util pentru afisarea textelor multilinie in browser:
Exemplu:

```
echo nl2br("text pe\n2 linii.");
```

Afiseaza:
text pe
2 linii.

- printf() si sprintf() – scrie un string sau parametru formatat. Diferenta e ca prima functie afiseaza direct in buffer, iar a doua intr-un string.
Caracterere de control sint:
d = nr. intreg
b = nr. intreb binar
x = nr. intreg haxazecimal
s = string
f = nr. real (float)
Exemplu:

```
printf("Nr: %d, cu 0 in fata: %04d, binar: %b, hexa: %x,
%s", 100,100,100,100, "[terminat]");
```

Afiseaza:
Nr: 100, cu 0 in fata: 0100, binar: 1100100, hexa: 64, [terminat]

- number_format() – formateaza numere, cu numar dat de zecimale sau caractere pentru separarea miilor si punctului zecimal.

- urlencode() si urldecode() – codifica un text oarecare cu caractere permise in adresele site-urilor. Multe din caractere nu sint permise in adresele site-urilor. De exemplu ‘(‘ va deveni %28, ‘)’ va deveni %29, “ “ va deveni %20, s.a.m.d.

- gethostbyname() si gethostbyaddr() – determina IP-ul unui calculator bazat pe numele domeniului, sau invers, obtine numele

dintr-un IP. Aveti grija la folosirea acestor functii pentru ca pot necesita destul de mult timp pentru executie.
Exemplu:

```
echo gethostbyname('www.example.com');
echo gethostbyaddr($_SERVER['REMOTE_ADDR']);
```

- mail() – trimite un email
Exemplu:

```
mail("joecool@example.com", "My Subject", "Line 1\nLine 2\nLine 3");
```