Cuprins

8. Scripturi server 3
8.1. CGI 3
8.2. SSI 3
8.3. ASP 4
8.4. PHP 5
8.5. Servere de Web 6
9. PHP avansat 6
9.1. Introducere 6
9.2. Php web server 7
9.3. Crearea unui script PHP 7
9.4. Executarea unui script PHP 10
9.3. Operatori 10
9.3.1. Operatorii aritmetici 10
9.3.2. Operatori logici 10
9.3.3. Operatori de incremetare-decrementare 10
9.3.4. Operatori de comparare 10
9.4. Variabile de mediu şi variabile definite de utilizator 11
9.4.1. Variabile de mediu 11
9.4.2. Variabile definite de utilizator 13
9.5. Tipuri de variabile 13
9.6. Şiruri de caractere 15
9.6.1. Includerea variabilelor în cadrul unui string 15
9.6.2. Conversia variabilelor la tipul string 16
9.6.3. Conversia variabilelor string la alte tipuri 16
9.6.4. Utilizarea şirurilor de caractere în parcurgerea tablourilor 16
9.7. Instrucţiuni condiţionale 17
9.7.1. If, else, elseif 17
9.7.2. While 17
9.7.3. Do... while 18
9.7.4. For 18
9.8. Require şi include 18
9.9. Funcţii definite de utilizator 19
9.10. Funcţii predefinite 19
9.10.1. Funcţii pentru manipularea şirurilor de caractere 19

9.10.2. Funcţii pentru manipularea tablourilor 20
9.10.3. Funcţii matematice 23
9.10.4. Funcţii pentru transmiterea unui mail direct din script 23
9.10.5. Funcţii pentru manipularea datei calendaristice 24
9.11. Clase si obiecte 24
9.12. Sesiuni de lucru 25
9.13. Cookie 26
9.14. Preluarea datelor din formularele HTML 27
9.14.1. Funcţii şi variabile pentru preluarea datelor din formulare 30
9.14.2. Funcţii destinate accesului la bazele de date relaţionale 30
9.15. Lucrul cu fişiere 37
9.16. Mesaje de eroare 38
9.17. Exerciţii PHP 39

8. Scripturi server

8.1. CGI

CGI (Common Gateway Interface) este un mecanism prin care clienţii (browserele) pot trimite informaţii înapoi la server, şi serverul poate trimite e-mail clientului cu datele cerute. De obicei, CGI se foloseşte pentru procesarea datelor dintr-un formular trimis de browser, şi se specifică în atributul „action" al unui formular.

Datele din formular sunt trimise la server la apăsarea butonului „Submit", sub forma unui şir de caractere: *numescript?numevar1=valoarevar1&numevar2=valoarevar2.*

În exemplul de mai sus, se trimite serverului numele scriptului CGI care trebuie executat (specificat în atributul „action" al formularului), împreună cu numele variabilelor şi valorile lor din formular. Scriptul de pe server le primeşte şi le procesează, apoi trimite un mail utilizatorului cu rezultatele. În esenţă, scriptul CGI trebuie să facă trei lucruri:

- să interpreteze şirul primit ca parametru, şi să scoată din el variabilele şi valorile lor (dacă este vorba de datele dintr-un formular);
- să proceseze aceste date;
- să trimită e-mail la o adresă anume, cu datele rezultate din urma procesării, eventual să genereze o pagină de Web.

Scripturile CGI de pe un server se ţin, de obicei, în subdirectorul *cgi-bin*. Majoritatea scripturilor CGI la ora actuală de pe Internet sunt scrise în limbajul Perl, dar există scripturi CGI şi în C, Java („servlets") şi alte limbaje. Desigur, serverul de Web trebuie să ofere suport pentru CGI.

Puteţi găsi multe scripturi CGI şi documentaţie la adresele http://www.cgiextremes.com, http://www.cgi-resources.com şi altele.

8.2. SSI

Serverul poate genera dinamic, „din zbor", fişiere HTML pe care să le trimită clientului spre afişare, în funcţie de datele trimise/cerute de client. Acest lucru se poate realiza cu SSI („Server-Side Includes"). SSI este un limbaj prin care se poate crea dinamic conţinutul paginii de Web trimis la client. Nu toate serverele de Web oferă suport pentru SSI.

De exemplu, pentru a include un antet la fiecare pagină afişată de browser, puteţi scrie în fişierul HTML, în corp, la început:

Exemplu:

```
<pre>
<!--#include file="antet.txt" -->
</pre>
```

Observaţi sintaxa comenzii SSI: <!--#include file="antet.txt" -->. Când un client cere această pagină, serverul procesează comanda şi trimite clientului fişierul HTML, dar în locul acestui comentariu (inclusiv „<" şi „>"), fişierul HTML va conţine textul din „antet.txt". Comenzile SSI sunt invizibile la client, deoarece clientul (browserul) primeşte doar rezultatul procesării acestor comenzi.

Exemplu*:*

```
  <!--#flastmod file="index.html" -->
```

Exemplul de mai sus va include în pagină data ultimei modificări ale fişierului „index.html”. Serverul, înainte de trimiterea fişierului HTML spre client, „se uită” la data ultimei modificări ale fişierului pe harddisc, şi include această dată în locul comentariilor. O variantă la exemplul de mai sus, pentru includerea datei ultimei modificări a fişierului în care este comanda SSI (fişierul curent):

```
<!--#echo var="LAST_MODIFIED" -->
```

Există şi alte comenzi SSI, puteţi să Fişierele HTML care conţin şi comenzi SSI au de obicei extensia „.shtml”; în acest fel, serverul nu trebuie să proceseze toate fişierele HTML pe care le trimite, doar cele cu această extensie specială.

8.3. ASP

ASP – Active Server Pages seamănă cu SSI, dar este mult mai complex. Este caracteristic serverelor de Web produse de Microsoft. Ca şi JavaScript şi VBScript, şi ASP înseamnă scripturi. Diferenţa este că în timp ce scripturile pe partea de client sunt trimise browserului client (de obicei înglobate în HTML), şi acesta le procesează local, ASP este procesat de către server, şi numai rezultatele sunt incluse şi trimise clientului în fişierul HTML. Fişierele de pe server care conţin cod ASP au de obicei extensia „.asp”, tocmai pentru a uşura munca serverului, ca acesta să nu proceseze degeaba toate fişierele care trebuie să le trimită. Putem folosi mai multe limbaje de script pentru ASP. Nu toate serverele suportă ASP.

Exemplu:

```
<html>
<body>
<%
response.write("Text de test.")
%>
</body>
</html>
```

Exemplul de mai sus este în VBScript (limbajul implicit pentru ASP). La cererea acestei pagini de către un client, serverul înlocuieşte marcajul <% %> şi ce conţine el cu textul „Text de test.” Observaţi felul cum sunt introduse comenzile ASP în sursa HTML – între „<%” şi „%>”. Clientul va primi:

```
<html>
<body>
Text de test.
</body>
</html>
```

Observaţi că clientul primeşte doar rezultatul procesării.

Exemplu:

```
<%@ language="javascript" %>
<html>
<body>
<%
Response.Write("Hello World!")
%>
</body>
</html>
```

Exemplul de mai sus face acelaşi lucru ca şi cel anterior, dar este scris în JavaScript. La începutul fişierului, se specifică „javascript” ca limbajul implicit pentru acest fişier. Dacă

lucrăm cu JavaScript, acesta face diferenţă între literele mari şi mici, nu ca VBScript. Se pot folosi şi alte limbaje de script, dar trebuie instalate interpretoarele aferente pe server (pe client nu este nevoie, deoarece acesta primeşte doar rezultatul).

8.4. PHP

PHP – original era „Personal HomePage”, apoi i s-a schimbat numele în „PHP: Hypertext Preprocessor”. Este un limbaj asemănător C-ului, şi se utilizează similar ASP-ului. Este dezvoltare Apache şi complet gratuit. Există interpretoare de PHP pentru multe servere de Web, chiar şi pentru servere produse de Microsoft.

De regulă, fişierele HTML care conţin cod PHP au extensia „.php”, sau variante („.php3”, „.php4”, după versiunea de PHP), ca serverul să le poată deosebi de fişierele HTML inactive (fără cod de program care trebuie procesat de server).

Exemplu:

```
<html>
<body>
<?php
echo "Text de test.<br>";
?>
</body>
</html>
```

Exemplul de mai sus scrie în fişierul HTML „Text de test”, apoi
. Observaţi modul introducerii comenzii PHP în sursa HTML, între „<?php” şi „?>”. Se poate omite „php”.

Exemplul următor vă prezintă un formular dintr-o pagină de web, şi pagina PHP care este accesată la apăsarea butonului Submit:

Formularul din pagina HTML:

```
<form action="raspuns.php" method="post">
Valoarea de trimis <input type="text" name="val">
<input type="submit">
</form>
```

Fişierul „raspuns.php”:

```
<html>
<head>
     <title>Pagina de raspuns</title>
</head>

<body>

Valoarea primita este <?php echo $_POST['val']; ?>.

</body>
</html>
```

La apăsarea butonului Submit în formularul din pagina HTML, browserul trimite „raspuns.php?val=textul” serverului, unde „textul” este textul scris de utilizator în căsuţa de text. Serverul procesează pagina „raspuns.php”, folosind datele returnate, adică valoarea câmpului „val” din vectorul „_POST”; pagina trimisă înapoi clientului va conţine „Valoarea primita este textul”, unde „textul” este valoarea lui „val”, adică textul pe care l-a introdus utilizatorul în căsuţa de text din formular.

Găsiţi documentaţie despre PHP la http://www.php.net sau alte situri.

8.5. Servere de Web

Serverele de Web sunt programe care rulează pe un calculator, ascultă pentru şi servesc cererile clienţilor. Cererile clienţilor înseamnă că un browser de pe un calculator din Internet se conectează la calculatorul pe care rulează serverul de Web, şi îi cere un fişier HTML. Serverul ia fişierul de pe harddisc, eventual îl procesează, şi îl trimite clientului, care apoi îl procesează şi îl afişează. Fişierul HTML implicit este de obicei „index.html”; de obicei, când vă conectaţi la un site, de exemplu http://www.utcluj.ro, serverul de acolo trimite implicit fişierul „index.html”.

Astfel de programe există în mai multe variante, chiar şi gratuite, pe Internet. Unele sunt mai simple, altele mai complexe; unele suportă SSI, ASP şi/sau PHP, altele nu. Unele sunt mai configurabile, altele mai puţin.

Un server de Web gratuit pentru Windows 95/98 şi Windows NT 4 este PWS, Personal Web Server. Acesta este inclus în pachetul de instalare Windows 98, în subdirectorul PWS din pachet. Pentru Windows 95 şi Windows NT 4, puteţi să aduceţi pachetul Windows NT 4 Option Pack de pe situl Microsoft (http://www.microsoft.com/msdownload/ntoptionpack/askwiz.asp), şi să instalaţi PWS din el. PWS ştie să lucreze cu pagini ASP şi SSI.

IIS – Internet Information Server este serverul de Web standard pentru Windows 2000/XP/2003/Vista, sau se poate instala separat pentru Windows NT Server 4.0. Este foarte puternic şi configurabil, cu suport ASP şi SSI înglobat.

Un alt server performant, gratuit şi standard pentru serverele Linux este Apache. Puteţi să-l aduceţi (chiar şi codul sursă al serverului) de la http://www.apache.org. Apache are şi versiune Windows. Este rapid, şi suportă SSI, precum şi PHP.

Problema cu un server de Web propriu este că serverul – şi calculatorul pe care rulează, precum şi legătura la Internet – trebuie să fie activă 24/7, adică tot timpul, zi şi noapte. O soluţie mai ieftină ar fi să vă plasaţi fişierele HTML care compun situl, pe un server dedicat, comercial sau gratuit.

Sunt câteva IPP-uri („Internet Presence Provider”) comerciale, cum ar fi UPC (http://www.upc.ro), RDS (http://www.rdsnet.ro) şi altele. Puteţi afla serviciile oferite, condiţiile şi preţurile acestor firme de pe siturile lor. Vizitaţi şi pagina Asociaţiei Naţională a Furnizorilor de Servicii Internet din România (http://www.anisp.ro).

Gratuite sunt şi mai multe, unele dintre cele mai cunoscute – internaţionale – fiind Geocities (http://www.geocities.com) şi Homestead (http://www.homestead.com).

Situri româneşti care găzduiesc gratuit paginile dvs. de HTML ar fi, de exemplu, K.ro (http://www.k.ro), Home.ro (http://www.home.ro), şi altele.

Problema cu siturile gratuite este că ele pun publicitate nedorită pe pagina dvs. – trebuie să trăiască şi ele din ceva.

9. PHP avansat

9.1. Introducere

Avantajul codului PHP îl reprezintă posibilitatea intercalării sale cu marcajele HTML, limbajul specific paginilor de internet. Condiţia este ca fişierele ce conţin cod PHP să fie salvate cu extensia .php.

Scrierea codului se realizează fie în editoare simple de text, fie în programe specializate. Pentru ca fişierele php să fie vizualizate de pe calculatorul personal, nu este necesară instalarea prealabilă a pachetului PHP şi a unui server de internet, acestea fiind instalate implicit în FedoraCore 5.

Fişierele php se execută prin încărcarea lor într-un browser de web.

Codul php poate intercepta valorile câmpurilor de intrare ale codului HTML.

9.2. Php web server

Versiunea 5.1 a PHP este inclusă in Fedora Core 5. Această versiune prezintă îmbunătăţiri faţă de versiunea anterioară printre care performanţe mai bune şi un modul nou pentru bazele de date PDO.

Au fost adăugate următoarele module:

- `date`, `hash`, şi `Reflection` (în cadrul pachetului `php`)
- `pdo` şi `pdo_psqlite` (în pachetul `php-pdo`)
- `pdo_mysql` (în pachetul `php-mysql`)
- `pdo_pgsql` (în pachetul `php-pgsql`)
- `pdo_odbc` (în pachetul `php-odbc`)
- `xmlreader` şi `xmlwriter` (în pachetul `php-xml`)

iar altele au fost eliminate:

- `dbx`
- `dio`
- `yp`

9.3. Crearea unui script PHP

Codul PHP se scrie intercalat sau separat de codul HTML al paginii web.

Codul HTML permite realizarea sub formă prietenoasă a interfeţei grafice cu utilizatorul. O pagină HTML are următorul format standard:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN">
<html>
<head>
<title>REAMINTIM</title>
<meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1">
</head>
<body>
</body>
</html>
```

În partea de HEAD se păstrează informaţii legate de titlul paginii, iar partea de BODY este cea în care se implementează instrucţiunile (*tag-urile*) HTML din spatele graficii: butoane, imagini, link-uri, etc. Codul php poate fi plasat între tag-urile </head> şi <body>, în secţiunea de BODY sau înaintea tag-ului HTML limitat în ambele situaţii de marcajele „<?” şi „?>” sau „<?php” şi „?>”

Comentariile se marchează prin „//” pentru o singură linie şi cu „ /*” şi „*/” pentru un comentariu ce se extinde pe mai multe linii.

Sfârşitul liniilor de cod de marchează cu „ ; ”.

Afişarea pe ecran a textului se poate realiza prin intermediul lui „echo” urmat de textul de afişat între ghilimele.

Exemplul 1:

```
<?php echo ” Un prim exemplu”; ?>
```

Codul se salvează într-un fişier cu extensia .php şi se încărcă într-un browser de web. Ca rezultat, pe pagina de web va apărea tipărit textul : Un prim exemplu .

Alte funcţii care permit transmiterea spre consolă a unor şiruri de caractere sunt:

Numele funcţiei	**Efect**	**Exemple**
print()	Tipăreşte pe ecran un şir de caractere	<? print(” sirul nostru este acesta”); print ” dar functioneaza si asa”; ?>

printf()	Oferă posibilitatea tipăririi formatate a unui şir de caractere; seamănă foarte mult cu funcţia printf() din limbjaul de programare C/C++. Formatările pot fi:
	%b - argumentul furnizat este un întreg şi va tipărit ca număr binar;
	%c - argumentul furnizat este un întreg şi va fi tiparit caracterul asociat acestui număr din codul ASCII;
	%d - argumentul este tratat ca un întreg şi prezentat ca şi un număr zecimal cu semn;
	%e – numărul scris în forma ştiinţifică
	%f- număr în virgulă flotantă;
	%o – numărul va fi tipărit în baza opt;
	%s - permite tipărirea unui şir de caractere;
	%x- numărul este tiparit cu litere mici în hexazecimal(baza 16 adică cu cifre între 0…9 , a…f);
	caracterul “_ “poziţionat după %
		<?
		$ex1 = 43951789;
		$ex2 = -439.51789;
		$ex3 = 3;
		$ex4=65;
		
		printf("
		reprezentarea binara anumarului %d este = '%b'\n", $ex3,$ex3);
		printf("
		printeaza caracterul avand codul ASCII %d = '%c'", $ex4, $ex4);
		printf("
		 pentru tiparirea intregilor = '%d'", $ex2);
		printf("
		 pentru tiparirea numerelor in forma stiintifica = '%e'", $ex1);
		printf("
		 pentru tiparirea unui numar in virgula flotanta = '%f'\n", $ex2);
		printf("
		 pentru tiparirea unui sir de caractere = '%s'", $ex1);
		?>
		
		<?
		$ex5 = 'exemple';
		$ex6 = "exemplele sunt multe:(";
		printf("
		iesirea standard a sirului de caractere:%s ",$ex5);
		printf("
		 aliniere la dreapta:%14s", $ex5);
		printf("
		aliniere la stanga cu caractere albe:%-13s", $ex5);
		printf("
		 adaugarea de zerouri la sirul de caractere:%010s", $ex5);
		printf("
		afisarea a doar 14

	perminte alinierea la dreapta a textului de tipărit; un număr după caracterul % ne spune câte caractere trebuie sa aibă şirul tipărit; caracterul “. “ urmat de un număr ne precizează cât de lung va fi şirul tipărit din lungimea totală.	caractere din totalul de %d:%10.14s", strlen($ex6),$ex6); ?>
sprintf()	Oferă posibilitatea formatării unui şir de carcatere şi apoi a tipăririi acestuia;vezi printf.	
vprintf()	Permite tipărirea unui tablou conform unei formatări specificate ca argument.	<? $date= array("data nasterii", "13/01/1970","cod numeric personal", 1700113123456,"CI Seria","MM","numarul",231234); vprintf(" Datele mele personale sunt %s: %s, %s: %14.0f, %s: %s, %s: %d.", $date); ?>
sscanf()	Preia dintr-un string valori conform unui şablon prestabilit.	<?php $ex7 = "tipareste pe ecran varsta: 36"; $n = sscanf($ex7, "%s %d", $text, $varsta); printf(" %s %d", $text, $varsta); $ex7 = "varsta: 36"; $n = sscanf($ex7, "%s %d", $text, $varsta); printf(" %s %d", $text, $varsta); ?>

fscanf(), fprintf()	Permite citirea sau scrierea unor texte la un terminal specificat conform unui format.	<?php
		$data=getdate();
		$fis = fopen('fisier.txt', 'w');
		
		//deschide un fisier pentru scriere,
		// il creeaza daca acesta nu exisita
		deja;
		
		fprintf($fis, "
		ultima
		modificare: %d %s %d",
		$data[year],$data[month],$data[mday]

		); ?>
flush()	Goleşte *buffer-ul* de ieşire forţând executarea comenzilor asociate.	

9.4. Executarea unui script PHP

Pentru aplicaţii pe calculatorul personal, codul PHP se poate crea în orice editor de text şi se salvează cu extensia .php în subdirectorul *htdocs* din directorul în care a fost instalat Xampp pentru LINUX pe PC; se înscrie apoi adresa sa, http://localhost//numefis.php , într-un browser de web.

De asemenea, fişierele php şi html pot fi stocate în directorul /var/www/html/ de unde pot fi accesate prin browser, de exemplu Mozzila Firefox.

9.3. Operatori

Operatorii limbajului php se clasifică în operatori aritmetici, de atribuire, pe bit, de comparare, de execuţie, de incrementare sau decrementare, operatori logici şi operatori de şiruri. În continuare, vor fi prezenataţi cei mai des utilizaţi.

9.3.1. Operatorii aritmetici

\+ adunare,

\- scădere,

* înmulţire,

/ împărţire întreagă,

% restul împărţirii.

9.3.2. Operatori logici

and	şi
or	sau.
xor	sau exclusiv.
!	operatorul de negare
&&	şi
||	sau

9.3.3. Operatori de incremetare-decrementare

++$a	incrementare anterioară utilizării variabilei „a” într-o instrucţiune,
$a++	incrementare ulterioară utilizării variabilei,
--$a	decrementare anterioară utilizării variabilei,
$a--	decrementare ulterioară utilizării lui a.

9.3.4. Operatori de comparare

= =	rezultatul este adevărat în caz de egalitate între membrul drept şi membrul stâng;

!=	rezultatul are valoarea logică adevărat dacă membrul drept diferit de membrul stâng;
<	rezultatul este adevărat dacă membrul drept mai mic decât membrul stâng;
>	rezultatul este adevărat dacă membrul drept mai mare decât membrul stâng;
<=	rezultatul este adevărat dacă membrul drept mai mic sau egal cu membrul stâng;
>=	rezultatul este adevărat dacă membrul drept mai mare sau egal cu membrul stâng.

9.4. Variabile de mediu şi variabile definite de utilizator

9.4.1. Variabile de mediu

Variabilele de mediu sunt variabile predefinite ale limbajului PHP şi au un format specific. În continuare, sunt prezentate câteva dintre ele:

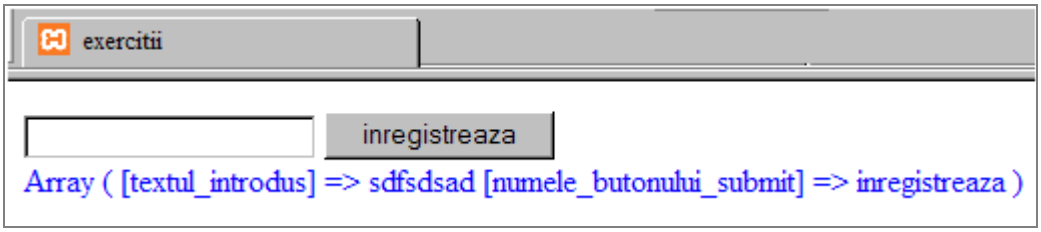
- **PHP_SELF:** păstrează fişierul php care se execută la momentul apelării sale, relativ la directorul rădăcină. Dacă fişierul php rulează ca o comandă de la linia de comenzi , variabila nu este disponibilă;
- **CONTENT_LENGTH** : păstrează lungimea, în octeţi, a corpului cererii HTTP(a browser-ului)) care a solicitat execuţia PHP.
- **CONTENT_TYPE** : păstrează tipul MIME(componenta fundamentală a protocolului de comunicaţie HTTP) al datelor din corpul cererii.
- **DOCUMENT_ROOT**: păstrează calea care constituie rădăcina arborelui catalogului cu documente al serverului Web.
- **HTTP_CONNECTION**: păstrează conţinutul antetului HTTP Connection, care indică opţiunile solicitate de client.
- **HTTP_HOST**: păstrează conţinutul antetului HTTP Host care indică numele de gazdă, folosit de client la prezentarea cererii.
- **HTTP_REFERER**: păstrează adresa URL a paginii Web care a trimis spre browser clientului la pagina curenta.
- **PATH** : calea de execuţie.
- **REMOTE_ADDR**: păstrează adresa IP a clientului.
- **REMOTE_HOST**: păstrează numele de gazda al clientului.
- **REMOTE_PORT**: păstrează adresa portului clientului de unde a pornit cererea.
- **REQUEST_METHOD**: păstrează metoda de cerere HTTP folosită; de exemplu, GET, POST, PUT sau HEAD.
- **REQUEST_URI**: păstrează URI folosit pentru accesul la pagina curentă. URI este alcătuit dintr-un URL şi un şir opţional de interogare.
- **SCRIPT_FILENAME**: păstrează numele de cale absolut al script-ului curent.
- **SCRIPT_NAME**: păstrează adresa URL a script-ului curent.
- **SERVER_HOST**: păstrează numele de gazdă asociat serverului Web care prelucrează cererea.
- **SERVER_PROTOCOL**: păstrează numele şi versiunea protocolului prin intermediul căruia s-a executat cererea.

- **SERVER_SIGNATURE**: păstrează şirul care identifică versiunea serverului Web şi numele de gazdă folosit pentru prelucrarea cererii.
- **SERVER_SOFTWARE**: şirul care identifică programul server Web şi versiunea acestuia.
- **HTTP_COOKIE_VARS:** reprezintă un şir de variabile asociate script-ului curent prin cookie-uri HTTP. Este disponibilă prin directiva `<? php_track_vars ?>`
- **HTTP_GET_VARS:** reprezintă un şir de variabile asociate script-ului curent. Variabilele sunt transmise prin intermediul metodei HTTP, GET. Este disponibilă prin directiva `<?php php_track_vars ?>` sau prin activarea variabilei de urmărire; în prezent se foloseşte $_GET.
- **HTTP_POST_VARS:** este un şir de variabile asociate script-ului curent şi transmis prin intermediul metodei POST a formularelor; în prezent se foloseşte $_POST având aceeaşi funcţionalitate.

Exemplul 1:

```
<html>
<head>
<title>exercitii</title>
<meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1">
</head>
<body>
<form method="POST">
<input type="text" name="textul introdus">
  <?php echo $HTTP_POST_VAR['test']; ?>
  <input type="submit" name="numele_butonului_submit"
value="inregistreaza">
<font color="blue">
<br>
<?php print_r($_POST);?>
</font>
</form>
</body>
</html>
```

Efectul este:



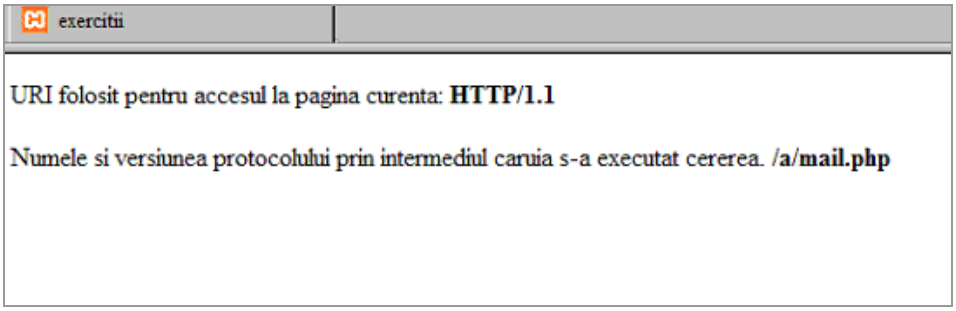
Exemplul 2:

```
<?php
$serv = $_SERVER['SERVER_PROTOCOL'];
echo "\nURI folosit pentru accesul la pagina curenta:
<b>$serv</b>";

$serv = $_SERVER['REQUEST_URI'];
```

```
echo "<p>Numele si versiunea protocolului prin intermediul
caruia s-a executat cererea. <b>$serv</b>"; ?>
```

Efectul:



9.4.2. Variabile definite de utilizator

Aceste variabile, la fel ca şi cele predefinite, sunt precedate de caracterul „&”. &nume='tot';//variabila şir de caractere

Variabilele definite într-un script sunt recunoscute în toate funcţiile ulterior definite în acelaşi fişier şi în toate fişierele incluse sau solicitate de script-ul respectiv.

9.5. Tipuri de variabile

Tipul variabilei	Mulţimea de valori	Observaţii şi exemple
boolean	Exprima o valoare de adevăr: TRUE(adevărat) sau FALSE(fals)	Se obţine cu ajutorul operatorului cast (Boolean) sau (bool); orice valoare diferită de vid are valoarea de adevăr true si orice variabilă cu valoare vidă este echivalentă cu valaorea de adevăr false.

integer	{..., -2, -1, 0, 1, 2, ...}-valori întregi scrise în baza de numeraţie 8,10,16	Un număr se poate converti la întreg prin operatorul cast (int) sau (integer) sau cu ajutorul funcţiei intval()
		`<?`
		`$var =23; // număr întreg`
		`$var =027; // numărul 23 scris în baza 8`
		`$var = 0x17; //numărul 23 scris în baza 16`
		`$ex=(int)(true) ; // variabila ex va primi valaorea 1`
		`?>`

float	Număr cu virgulă mobilă	Pentru a converti un întreg în float
		`<?`
		`$ex1 = 2.3;`
		`$ex2 = 0.23e1;`
		`$ex3= 7E-10; //echivalent cu 0,0000000007`
		`?>`

string	O variabilă de tipul string păstrează şiruri de caractere	Poate fi incadrat de ghilimele sau aprostroafe.
		`<?`
		`$ex1 = `2.3`; //sirul de caractere avand 2 cifre si un punct`

		`$ex2 =”2.3”;//numarul 2.3`
		`$ex3= (string)($ex2);//variabila ex3 va pastra sirul de caracere corespunzator numarului real 2.3`
		`?>`

array	Tip de caractere ce permite păstrarea de tablouri uni şi multidimensionale de variabile	`<?`
		`$ex1=array(1=>2,2=>14, 5=>"incercare",6=>"true");`
		`echo "$ex1[2]"; //se va afisa 14`
		`?>`

object		`<?`
		`class exercitii`
		`{`
		`function ex1()`
		`{`
		`echo “este doar o incercare”;`
		`}`
		`}`
		
		`$unu = new exercitii;`
		`$unu->ex1();`
		`?>`

resource	Tip special de variabile păstrând referinţe la o resursă externă(fişiere deja deschise, conexiuni la baze de date)	Nu orice variabilă poate fi convertită la tipul *resource*.

null	Reprezintă o variabilă care nu are nici o valoare; are o singură valoare: NULL.	`<?`
		`$var=””;`
		`if (empty($var)==NULL) {echo “variabila nu retine o valoare”;}`
		`?>`

mixed	Indică faptul că un parametru poate accepta valori de tipuri diferite.	
number	Indică faptul că un parametru poate sa fie sau de tipul integer sau de tipul float.	

PHP nu solicită definirea explicită a tipului unei variabile la declararea acesteia. Tipul variabilei va fi determinat din contextul în care acea variabilă va fi folosită.

Exemplu:

```
$ex1=”23456”; //ex1 este variabila de tipul string
$ex1=23456; //ex1 este variabila de tipul integer
$ex2=23456 + "1 unitate"; //ex1 va avea valoarea  23457
```

De asemenea, pentru modificarea tipului variabilei se utilizează operatorul *cast* adică numele tipului la care variabila va fi convertit între paranteze rotunde.

Exemplu:

```
<?
$ex1=22;
$ex2=(string)$ex1;
?>
```

9.6. Şiruri de caractere

Un string este un şir de caractere. Există 256 de valori diferite. Şirurile de caractere sunt marcate de ghilimele(””) sau de apostroafe(’ ’). Din acest motiv, pentru a tipări literalul ’, este necesară precedarea sa de un backslash(\). În consecinţă, dacă este necesar ca (\) sa apară înaintea ultimului apostrof care marchează sfârşitul unui şir de caractere, acest (\) va fi dublat de încă unul (\). Când se utilizează ghilimelele, se pot folosi caracterele speciale evidenţiate în tabelul de mai jos.

Caractere speciale	Efect
\n	Linie nouă
\r	carriage return
\t	tab orizontal
\\	backslash
\$	Semnul $
\"	ghilimele

9.6.1. Includerea variabilelor în cadrul unui string

Dacă sunt plasate în interiorul unui şir de caractere, variabilele vor fi percepute ca atare. Variabilele pot fi identificate simplu prin simbolul $ urmat de numele variabilei sau acelaşi simbol urmat de numele variabilei între acolade {}(începând cu PHP4). Această ultimă variantă e necesară dacă se doreşte completarea numelui variabilei explicit sau prin introducerea unor expresii complexe.

Exemplu:

```
<?
$oras = 'oraş';
echo "Cluj-Napoca este un $oras\n ";
echo "${Oraş}ul in care ne aflam nu se numeşte Cluj-Napoca";
?>
```

Următorul exemplu prezintă utilizarea operatorului de concatenare a doua şiruri.

Exemplu:

```
<?
$oras = 'Bistrita';
echo "variabila are lungimea".   strlen($oras);//concatenarea
se realizează cu ajutorul
// simbolului  “.”
echo "\n prima litera este $oras[0] ";?>
```

9.6.2. Conversia variabilelor la tipul string

Se poate realiza cu ajutorul operatorului cast (string) sau cu funcţia strval().

Exemplu:

```
<?
 $ex1=TRUE ;
 $ex2=FALSE;
 $ex3=(string) $ex1;
 $ex4=(string) $ex2;
 $ex5=1.17e2;
 $ex6=(string) $ex5;
 $ex7=array("a" => "1",2 =>17);
 $ex8=(string) $ex7;
echo " valoarea de adevar TRUE este egala cu sirul de
caractere $ex3\n" ;
echo "  valoarea de adevar FALSE este egala cu sirul de
caractere $ex4\n    " ;
echo "\n valoarea float   $ex5 egala cu sirul de caractere \"
$ex6 \" " ;
echo "\n prin conversia unui tablou avand ca elemente  $ex7[a]
si $ex7[2]   la sir de caractere se obtine valoarea \"$ex8 \""
;
?>
```

9.6.3. Conversia variabilelor string la alte tipuri

Şirurile de caractere care conţin ca prime caractere cifre pot fi convertite la tipuri de date numerice. Dacă primele caractere nu sunt numere atunci valoarea obţinută prin conversie va fi 0.

Exemplu:

```
<?
$ex1 = (integer)"23";
echo "  (integer) \"23 \" = > numarul intreg $ex1 \n" ;
$ex1 = 1 + "-1.3e3";
echo "  \" 1 + \"-1.3e3\" \" = > numarul real $ex1 \n" ;

$ex1 = (real)"anul 2006";
echo " (real) \" anul 2006 \" = > numarul real $ex1 \n" ;
$ex1 = "28.02 are doua zecimale" + 12;
echo " \" 28.02 are doua zecimale anul 2006 \"+ 12 = > numarul
real $ex1 \n" ;

?>
```

9.6.4. Utilizarea şirurilor de caractere în parcurgerea tablourilor

În exemplul de mai jos , şirul de caractere “rosu” este de fapt “ poziţia” la care se regăseşte valoarea “negru” în tabloul dat.

Exemplu:

```
<?php
$ex1 = array("rosu" => "negru", "flori"  =>25, 2006 => TRUE);
echo " $ex1[rosu]"; // fara ghilimele
echo " $ex1[flori]";
echo " $ex1[2006] ";

$ex2=$ex1["rosu"];//cu ghilimele
?>
```

9.7. Instrucţiuni condiţionale

9.7.1. If, else, elseif

În funcţie de situaţie, sintaxa aplicabilă este una din următoarele:

if (expresie) instrucţiune;

sau

if (expresie) {intructiune1; instrucţiune2;...}

sau

if (expresie1) instrucţiune1;
else intrucţiune2;

sau

if (expresie1) instrucţiune1;
elseif (expresi2) instrucţiune2;
elseif (expresie3) instrucţiune3;
else instrucţiune4;

Instrucţiunea este executată daca valoarea expresiei este *true*(adevărat).

Pentru *if, else, elseif* există şi o instrucţiunea alternativă care presupune eliminarea parantezelor acolade şi înlocuirea lor cu „*endif*" la finalul secţiunii de instrucţiuni.

Exemplu:

```
_if ($a<$b)  echo " $a este mai mică decat $b";
          $a=$b;
endif;
```

9.7.2. While

WHILE este o instrucţiune de repetare în buclă cu condiţie iniţială şi are sintaxa:

while (expresie) instrucţiune;

sau

while (expresie) {intructiune1; instrucţiune2;...}

Instrucţiunea se execută repetat atât timp cât valoarea expresiei este *true*(adevărată). Dacă valoarea iniţială a expresiei este diferită de *true(adevărat)* atunci instrucţiunea nu se execută nici măcar o dată.

Instrucţiunea alternativă este:

while (expresie) instrucţiuni;...

endwhile;

9.7.3. Do... while

Sintaxă:

Do instructiune;

While (expresie);

Instrucţiunea se execută cel puţin o dată până la evaluarea expresiei sau de mai multe ori dacă valoarea de adevăr a expresiei este *true.* Execuţia buclei poate fi întreruptă prin utilizarea instrucţiunii de salt *BREAK* care va determina încheierea imediată a buclei şi continuarea programului cu instrucţiunile imediat următoare.

9.7.4. For

Sintaxa:

for (expr1; expr2; expr3) instrucţiuni;

La începutul buclei *for*, prima expresie evaluată este *expr1*.

Expr2 se evaluează la începutul fiecărei parcurgeri a buclei. Numai dacă valoarea de adevăr este *true*, instrucţiunile din corpul buclei sunt efectuate; altfel se iese din buclă.

Expr3 se execută la sfârşitul fiecărei iteraţii.

Sintaxa pentru instrucţiunea alternativă este:

for (expr1; expr2; expr3) instructiuni; ...;

endfor;

9.8. Require şi include

Cu ajutorul celor două instrucţiuni, *require* şi *include*, conţinutul fişierului specificat ca şi parametru este inclus în cadrul unui anumit script din interiorul căruia este executat. Toate variabilele din fişierul apelant sunt recunoscute de fişierul apelat.

Include va citi în interiorul fişierului inclus chiar dacă instrucţiunea nu este executată niciodată. Structurile în buclă nu afectează comportamentul său care se va executa o singură dată chiar dacă bucla se execută de mai multe ori.

Require va citi fişierul de inclus şi, dacă nu există eroare la executarea instrucţiunilor corespunzătoare, va continua executarea întregului cod.

Dacă cele două instrucţiuni sunt folosite în interiorul unor structuri de control, ele vor fi încadrate de paranteze acolade.

Exemplu:

Fişierul exa.php:

```
<?php
```

```
$nr1=12;
$nr2=13;
?>
```

Fişierul exb.php:

```
<?php
$s=$nr1+$nr2;
print "Suma celor doua este:".$s;

require 'exa.php';$s=$nr1+$nr2;
print "<br>Suma celor doua este:".$s;
?>
```

9.9. Funcţii definite de utilizator

Pentru a defini funcţii PHP trebuie respectată sintaxa:

function *nume_funcţie($argumen2, $argument2,...){*

Instrucţiune1;

Instruţiune2;...

return *valoare;*

}

Apelarea funcţiei:

$a=nume_funcţie(val1,val2,....);

Pentru ca argumetele funcţiei să fie transmise prin referinţă astfel încât modificările suferite în interiorul funcţiei să fie recunoscute şi în exteriorul ei, argumetul se declară de forma: &$variabila.

Exemplu:

```
 function  referintă(&$var) {
                $var=3;
     }
     $ex = 0;
     referinta($ex);
                echo ”Rezultatul este $ex”;    // Rezultatul
este  3
```

În PHP, există conceptul de variabilă funcţie. Dacă la un nume de variabilă sunt ataşate paranteze rotunde, PHP va căuta o funcţie cu acelaşi nume şi va încerca să o execute.

9.10. Funcţii predefinite

Sunt funcţii puse la dispoziţie de mediul PHP pentru manipularea datelor.

9.10.1. Funcţii pentru manipularea şirurilor de caractere

Numele funcţiei	Efect	Exemple (unde este cazul)
addslashes()	Adaugă backslash in faţa caracterelor specificate.	
chr()	Transformă codul ASCII	

	în caracterul asociat.	
explode()	Transformă un şir de caractere într-un tablou.	
implode()	Transformă elementele unui tablou într-un şir de caractere.	
ltrim()	Elimină spaţiile din capătul stâng al şirului de caractere.	
rtrim()	Elimină spaţiile din capătul drept al şirului de caractere.	

sscanf()	Citeşte dintr-un şir de caractere valorile corespunzătoare unor variabile pe baza unui format.	Exemplul de mai jos va genera afişarea tipului variabilelor şi a valorii corespunzatoare:
		`<?`
		`$sir="capitol php ora 1";`
		`sscanf($sir,"capitol %s ora %d",$cap,$ora);`
		`var_dump($cap,$ora);`
		`?>`

strcmp()	Compară două şiruri de caractere.	
strstr()	Găseşte prima apariţie a unui şir într-un alt şir de caractere.	
substr()	Returnează un subşir dintr-un şir de la o poziţie specificată şi conţinând un număr dat de caractere.	`<?php` `echo " ".substr('sirul de test', 1);` `echo " ".substr('sirul de test', 0, 4);` `echo " ".substr('sirul de test', -1, 1);` `$ex= 'sirul de test';` `echo $ex{3};` `?>`
substr_compare()	Compară două şiruri de caractere specificând o anumită poziţie de start.	
strlen()	Returnează lungimea şirului de caractere.	`<?` `$sir="capitol php ora 1";` `print_r("  sirul meu \" $sir\" are lungimea de". strlen($sir)."caractere" );` `?>`

9.10.2. Funcţii pentru manipularea tablourilor

Aceste funcţii permit manipularea tablourilor, fiind des folosite la stocarea, modificarea şi operarea asupra diferitelor tipuri de date.

Există tablouri cu una sau cu mai multe dimensiuni şi pot fi create de utilizator sau de către alte funcţii.

Numele funcţiei	**Efect**	**Exemplu**
array(cheie=>valoare,...)	Creează un tablou; elementele tablourilor nu trebuie sa fie de acelaşi tip sau să respecte succesiunea imediată	`<?` `$ex8= array("html" => 1, 5=>"php");` `$ex88= array(1, "php");` `print " $ex8[html], $ex8[5]";` `print " $ex88[0], $ex88[1]";` `?>`
array_change_key_case(sirul,tip)	Modifică un şir la majuscule sau litere mici. Acest tip poate avea valorile CASE_LOWER si CASE_UPPER	`<?` `$ex8= array("html" => 1, 2 =>"php");` `echo " array_change_key_case($ex8, CASE_UPPER)";` `?>`
array_chunk(tabloul,p,cheie)	Împarte un tablou în mai multe sub-tablouri, fiecare cu o lungime dată de parametrul p; dacă cheie este *true* atunci se păstrează cheile tabloului iniţial.	
array_combine(tablou1, tablou2)	Se obţine un nou tablou păstrând de la primul cheile şi de la al doilea valorile.	`<?` `$ex8= array("html", 2);` `$ex9=array(1,"php");` `$ex10= array_combine($ex8,$ex9);` `print " ";` `print_r($ex10);` `?>`
array_merge(şirul1, şirul2)	Se obţine un nou tablou prin alipirea elementelor celor două tablouri.	`<?` `$ex8= array("html", 2);` `$ex9=array(1,"php");` `$ex10= array_merge($ex8,$ex9);` `print " ";` `print_r($ex10);` `?>`

array_shift(sirul)	Elimină primul element dintr-un tablou şi returnează valoarea elementului eliminat.	`<?`
		`$ex8= array("html", 2);`
		`print_r(array_shift($ex8));`
		
		`?>`

in_sort()	Verifică dacă un element specificat se află printre elementele tabloului.	
rsort()	Sortează un tablou in ordine descrescătoare.	`<?` `$ex11= array(23,2,15,6);` `rsort($ex11);` `foreach ($ex11 as $i => $nr) {` `print " "."ex11["` `. $i . "] = " . $nr;` `}` `?>`
sort()	Sortează crescător un tablou.	
sizeof()	Returnează numărul elementelor tabloului.	
reset()	Pointerul va indică primul element din şir.	
end()	Pointerul va indica spre ultimul element.	

each()	Returnează cheia curentă(indicele) şi valoarea asociată.	`<?`
		`$ex11= array(23,2,15,6);`
		`$date_sir = each($ex11);`
		`print_r($date_sir);`
		`$date_sir = each($ex11);`
		`print_r($date_sir);?>`

current()	Returnează valoarea elementului curent din tablou(spre care indica	`print_r(current ($ex11));`

	pointerul).	

9.10.3. Funcţii matematice

Permit diferite operaţii cu numere . Mai jos sunt prezentate doar câteva dintre ele.

Funcţia	Efectul	Exemplu
abs()	valoarea absolută a unui număr	
sin(),cos(),acos()atan(),asin(),...	funcţii trigonometrice	
exp()	Returnează valoarea lui e^x.	
floor()	Rotunjeşte un număr zecimal spre intregul mai mic cel mai apropiat.	
round()	Rotunjeşte un număr spre cel mai apropiat întreg.	
min(),max()	Minimul si maximul a două numere.	
sqrt()	Rădăcină pătrată a unui număr.	
rad2deg()	Converteşte un număr din radiani în grade.	

```
<?

$ex13=30;
$ex14=-25.6;
$ex15=23.89;
print
"<br>abs($ex14)=".abs($ex14)
;
print
"<br>sin($ex13)=".sin($ex13)
;
print
"<br>round($ex15)=".round($e
x15);
print
"<br>min($ex15,$ex14,$ex13)=
".min($ex15,$ex14,$ex13);
print
"<br>rad2deg(pi)=".rad2deg(p
i());
?>
```

9.10.4. Funcţii pentru transmiterea unui mail direct din script

Funcţia utilizată este:

mail(destinatia,subiectul, mesajul,header,parametri)

Funcţionarea depinde de setările existente în fişierul php care se află în */usr/local/lib/**php.ini*** sau */etc/**php.ini***. Aici se vor seta pentru categoria **[mail functions]** următoarele valori: *SMTP =mail_server_SMTP* adică serverul folosit pentru trimiterea mesajelor spre destinaţia stabilită în codul php, *smtp_port =SMTP_port* şi *sendmail_path = /usr/sbin/sendmail.*

Exemplu:

```
<?
$mesaj="primul mail\n Incercam sa trimitem un mail la curs";
$header="from:cont@ mail_server_SMTP.com";
```

```
$mesaj_final=wordwrap($mesaj,70);//daca liniile sunt mai lungi
de 70 de caractere
mail("cont@mail2.com ","exemplu",$mesaj_final,$header);
?>
```

9.10.5. Funcţii pentru manipularea datei calendaristice

Pentru oţinerea datei curente:

date(forma, timestamp)

forma - forma datei calendaristice(exemple ”y.m.d”, „”y/m/d”)

timestamp - numărul de secunde cuprins între 1 ianuarie 1970, ora 0:00 şi prezent

Pentru crearea unei valori de tip data calendaristică se utilizează funcţia: *mktime(ora,minutul,secunda,luna,yi,an,[is_dst]).*

Exemplu:

```
<?
$ex_data=date("y/m/d");
echo $ex_data;
$azi=mktime(11,11,0,10,5,2006);
echo "<br>data este". date("y/m/z",$azi);
?>
```

Se observă utilizarea variabilei $azi ca ştampilă de timp.

9.11. Clase si obiecte

O clasă este o colecţie de variabile şi funcţii. Sintaxa folosită în declararea unei clase este exemplificată mai jos:

```
     <?
                class  Max{
                          var $var1;
                          var $var2;
               function  iniţializare($var11, $var22) {
$this->var1=$var11;//variabila var1 primeste valoarea
variabilei var2
//transmisă ca argumet
                $this->var2=$var22;}
          function verificare ($var11, $var22)  {
if ($this->var1>$this->var2{ return false;} //dacă var1 este
mai mare ca var2          //atunci rezultatul întors de
funcţie //este  false
else { return true;}}
     }
          ?>
```

Pentru instanţierea unei variabile de tipul max se foloseşte operatorul **new.**

Exemplu:

```
$ex1 = new Max;
```

Apelarea uneia dintre funcţiile clasei se realizează astfel:

Exemplu:

```
 $ex1->verificare(12,13).
```

Rezultatul este fals pentru că 12 este mai mic decât 13.

Funcţiile şi variabilele unei clase pot fi preluate de o alta prin procedeul de extindere a clasei utilizând cuvântul cheie **extends**. Noua clasă îmbogăţeşte caracteristicile clasei extinse prin adăugare de noi funcţii şi varibile.Moştenirea multiplă nu este posibilă.

Exemplu:

```
class MaxMin extends Max
```

Constructorul unei clase este o funcţie cu acelaşi nume ca şi clasa care se aplează automat la instanţierea unui obiect. În cazul claselor derivate, constructorul clasei parinte nu este apelat automat la apelarea constructorului clasei.

9.12. Sesiuni de lucru

Sesiunile de lucru permit păstrarea pe server a unor date şi informaţii despre utilizator de-a lungul succesiunii de pagini parcurse de acesta. Fiecare solicitare de acces la o pagină este însoţită de identificatorul de sesiune care poate fi setat prin funcţia *session_start()* sau va fi preluat automat prin *session_request()*. Apelul funcţiei *session_start()* trebuie poziţionat înaintea marcajului <*html*> din pagina de internet. Mai jos este prezenentată utilizarea variabilei globale $_SESSION pentru asocierea de valori anumitor variabile tip sesiune.

Exemplu:

Se creează câte o sesiune de lucru în fiecare din următoarele fişiere:

```
//a.php
<?php
session_start();//startează o sesiune de lucru
$_SESSION['a']="prima pagina";//asociază variabilei a un sir
de caractere
echo '<br /><a href="b.php">legatura spre pagina urmatoare
</a>';
echo $_SESSION['b']; //tipareste valoarea variabilei b setata
in pagina anterioara
?>
```

şi

```
//b.php
<?php
session_start();
$_SESSION['b']="pagina a doua";
echo '<br /><a href="a.php"> legatura spre pagina anterioara
</a> ';
echo $_SESSION['a'];
?>
```

Alte funcţii care se folosesc asociat sesiunilor de lucru ale utilizatorului sunt descrise în tabelul următor.

Funcţia	Efectul	Exemple

unset(numele de variabilă session intre apostroafe sau ghilimele)	Şterge variabila specificată ca argument	`unset($_SESSION['a']);`
isset(numele de variabilă session intre apostroafe sau ghilimele)	Verifică dacă este setată variabila furnizată ca argument.	`isset($_SESSION['a']))`
session_destroy()	Şterge datele asociate sesiunii curente.	`session_destroy();`

session_encode()	Returnează un şir de caractere în care variabila furnizată va fi înglobată.	`<?`
		`session_start();`
		`$_SESSION['pag1']="prima pagina";`
		`$pag22 = session_encode();`
		`print_r($pag22);`
		`?>`

session_id()	Este folosită pentru a seta sau pentru a afla care este identificatorul sesiunii de lucru.	`<? session_start();`
		`echo session_id();`
		`session_destroy();`
		`?>`

SID(string)	Constanta păstrează numele şi identificatorul sesiunii sau nu are nici o valoare dacă imediat anterior a fost realizată o sesiune cookie.	
session_register (şiruri păstrând numele variabilelor)	Păstrează una sau mai multe variabile globale din sesiunea curentă. Verificaţi în fişierul php.ini dacă register_globals = on.	Nu mai este în uz.
session_name([numele sesiunii])	Setează sau returnează numele sesiunii de lucru	
session_regenerate_id ()	Actualizează identificatorul sesiunii cu unul nou generat.	
session_save_path ([şirul de carcatere care redă calea])	Returnează sau setează cale de salvare a sesiunii.	`$cale = session_save_path();`
session_write_close ()	Este utilizată la salvarea şi încheierea imediată a unei sesiuni de lucru.	

9.13. Cookie

Variabilele cookie sunt de fapt valori transmise clientului şi stocate pe calculatorul acestuia. Sunt utilizate cu preponderenţă pentru identificarea diferitelor situaţii de acces a siturilor de internet.

Setarea variabilei *cookie* se realizează prin funcţia *setcookie()*.

Exemplu:

```
setcookie("limbaj", $valoare, time()+36000);
```

Valoarea salvată pe harddisk este preluată cu ajutorul variabilei de sistem $_COOKIE.

Exemplu:

```
          $limb=$_COOKIE["limbaj"]; //atribuie valoarea
acestui cookie variabilei $limb
```

9.14. Preluarea datelor din formularele HTML

Cel mai adesea, programele PHP preiau datele furnizate de utilizator prin intermediul formularelor HTML. Cu ajutorul acestora, se implementează în pagină butoane radio, casuţe de tip text, parolă, suprafaţă de text, casuţe tip select, butoane de reset şi submit.

Formularele HTML definesc în principal 3 caracteristici importante: **action**, **method** şi **name.** *Name* identifică în mod unic fiecare formular, *action* specifică ce acţiune se va efectua prin intermediul formularului(în general, numele sau calea spre fişierul care va prelua valorile introduse în formular), *method* specifică metoda: *post* sau *get*.

POST permite preluarea datelor introduse în formular şi transmiterea lor spre un element de procesare.

GET permite preluarea datelor de la adresa specificată şi utilizarea lor în pagina curentă. Cantitatea de informaţii care poate fi transmisă este mult mai mică decât în cazul post.

Metoda folosită este importantă în special la comunicarea cu servere de baze de date: dacă scriptul trebuie să extragă date din baza de date şi să le afişeze în pagina de web de răspuns, se foloseşte metoda GET; dacă scriptul trebuie să insereze ceva în baza de date (de obicei, informaţii primite dintr-un formular html), se foloseşte metoda POST.

Formularele delimitează toate informaţiile care vor fi preluate de fişierul specificat prin *action.*

Exemplu „echoback”:

Sursa fişierului *formtest.html*	**Rezultat**
<html> <head> <title>Form test</title> </head> <body> <form action="echoback.php" method="post"> <input type="text" **name="casuta"**> <input type="submit"> </form> </body> </html>	
Sursa fişierului *echoback.php*	**Rezultat**
<html> <head> <title>Pagina de raspuns</title>	Variabilele primite sunt: casuta = tralala

Sursa fişierului *formtest.html*	**Rezultat**
</head> <body> Variabilele primite sunt: <?php foreach ($_POST as $key => $temp) echo $key." = ".$temp." "; ?> <body> <html>	

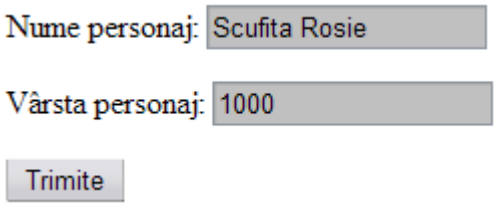
Exemplul de mai sus conţine un formular simplu cu o căsuţă de text, care are numele „casuta”. La apăsarea butonului, browserul „adună” informaţiile introduse de utilizator în formular, adică textul din căsuţă. Apoi, browserul trimite aceste informaţii la scriptul specificat de pe serverul specificat sau, în cazul în care serverul nu este specificat (ca în cazul nostru), aceluiaşi server de la care a primit şi fişierul cu formularul. Scriptul specificat, în cazul nostru, este „echoback.php”.

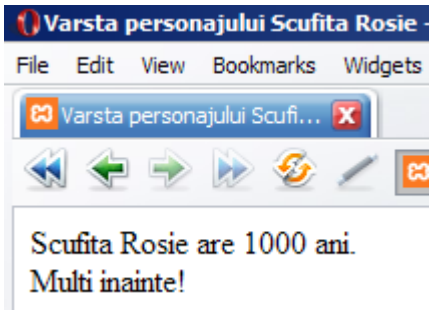
De fapt, browserul „cere” de la server pagina cu adresa „http://serverul/echoback.php?casuta=tralala”, prin această metodă transmiţând, de fapt, datele din formular, scriptului de pe server, pentru procesare. Dacă ar fi mai multe perechi *variabilă=valoare*, ele ar fi separate prin caracterul „&”.

Serverul primeşte cererea, porneşte scriptul „echoback.php” şi îi transmite variabila „casuta” cu valoarea „tralala”, ca şi argumente. Scriptul extrage, pe rând, perechile de nume/valoare variabilă din şirul $_POST. „$key” va fi numele variabilei, pe când „$temp” va fi valoarea ei.

Comanda „echo” nu afişează textul pe ecranul serverului, ci generează, din zbor, textul „fişierului” html ce va fi trimis înapoi browserului. Caracterul „.” (punct) este operatorul de concatenare de text, în PHP. Pentru o listă mai lizibilă a rezultatului, scriptul pune fiecare pereche de nume/valoare variabilă primită pe linii separate. Ca browserul să facă acest lucru, scriptul adaugă câte un „
” la sfârşitul fiecărei linii trimise în răspunsul html.

Exemplu „ani”:

Sursa fişierului *ani.html*	**Rezultat**
<html> <head> <title>Anii...</title> </head> <body> <form action="raspuns.php" method="post"> Nume personaj: <input type="text" name="nume"> Vârsta personaj: <input type="text" name="varsta"> <input type="submit" value="Trimite"> </form> </body> </html>	
Sursa fişierului *raspuns.php*	**Rezultat**

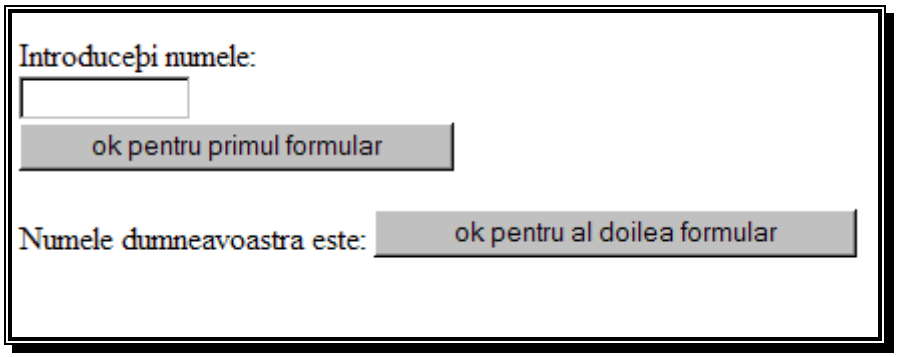
Sursa fişierului *ani.html*	Rezultat
<html> <head> <title>Varsta personajului **<?php echo $_POST['nume']; ?>**</title> </head> <body> **<?php echo $_POST['nume']; ?>** are **<?php echo $_POST['varsta']; ?>** ani. Multi inainte! </body> </html>	

Observaţi că scriptul PHP din exemplul de mai sus conţine trei blocuri de instrucţiuni PHP. Dacă am folosi un singur bloc, codul html dintre ele („</title>” şi celelalte) ar trebui generat cu comenzi „echo”, ceea ce ar fi ineficient. În schimb, am alternat cod html cu cod PHP. În aceste cazuri, trebuie să fim atenţi la deschiderea/închiderea blocurilor PHP.

Exemplu: Se consideră următorul formular HTML salvat sub numele de formular.htm:

```
<html>
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1">
</head>
<body>
<form name="exemplu1" action="ex1.php" method="POST">
Introduceti numele:<br>
<input type="text" name="text1"  size="10"><br>
<input name="ok1" type="submit" value="ok">
</form>
<form name="exemplu2" action="ex2.php" method="GET">
Numele dumneavoastra este:
<input name="ok2" type="submit" value="ok pentru primul
formular">
</form>
</body>
</html>
```

Încărcarea în browser va genera următorul rezultat:



În fişierul formular.html sunt definite două formulare diferite, la care atributul *method* este *post*, respectiv *get*. Se observă că numele butoanelor şi al formularelor diferă. Acţiunile pe care le vor executa formularele sunt definite de fişierele ex1.php şi ex2.php.

Fişierul ex1.php poate avea următoarea formă:

```
<?php
echo "numele tau este ".$_POST["text1"];
?>
```

iar fişierul ex2.php :

```
<?php
echo "Nu prea il putem ghici!!!!";
?>
```

La încărcarea în browser, utilizatorul va avea posibilitatea să tasteze un text în câmpul *text1* şi să trimită datele spre server dând click pe butonul „ok pentru primul formular".

În acel moment, datele vor fi preluate de fişierul ex1.php.

Scriptul php identifică fiecare câmp şi îi păstrează valoarea într-o variabilă. În funcţie de această valoare, scriptul php va efectua o anumită acţiune. Folosind o pagină html şi una php, în funcţie de acţiunea utilizatorului sau de nişte valori introduse pot fi generate ca răspuns 2 pagini diferite adică pagina de răspuns este creată **dinamic.**

Una dintre variante este mai specială şi evidenţiază capacitatea de comunicare şi intercalare a codurilor celor două limbaje.

9.14.1. Funcţii şi variabile pentru preluarea datelor din formulare

Funcţiile care permit verificarea introducerii valorilor în campurile specificate de către un formular sunt **isset()** şi **empty()**. Funcţia **isset()** verifică dacă câmpul a primit o valoare şi returneaza true în cay afirmativ, altfel returnează false şi are valoare implicită true. Funcţia **empty()** este true dacă nu camp nu a primit o valoare(dacă este egal cu "").

Pentru a accesa variabila dorită din cadrul formularului se folosesc variabilele predefinite **$_POST["numele variabilei"]** şi **$_GET["numele variabilei"]** în funcţie de metoda specificată în formular. În cazul variabilei $_GET, valoarea nu poate depăşi 100 caractere(vezi ultimul exemplu al capitolului).

Există o variabilă care poate fi folosită în ambele cazuri(atât cu post, cât şi cu get): **$_REQUEST.**

9.14.2. Funcţii destinate accesului la bazele de date relaţionale

PHP este capabil să interacţioneze cu baze de date relaţionale, un aspect esenţial pentru aplicaţiile actuale. Numeroasele funcţii puse la dispoziţie de limbaj fac posibil accesul variat la datele stocate.

Într-un prim pas este nevoie de stabilirea unei legături cu server-ul de baze de date şi cu sistemul de gestiune al bazei de date prin funcţiile*, mysql_connect* (realizează conectarea server de baze de date) şi *mysql_select_db*(selectează baza de date).

Exemplu „personaje":

Sursa fişierului *personaje.sql*

```
-- daca tabelul exista, il stergem
DROP TABLE IF EXISTS `personaje`;

-- cream structura tabelului
CREATE TABLE IF NOT EXISTS `personaje` (
  `nume` varchar(20) NOT NULL default '',
  `puteri` varchar(20) NOT NULL default '',
  `varsta` int(3) NOT NULL default '0'
);

-- inseram datele
INSERT INTO `personaje` (`nume`, `puteri`, `varsta`) VALUES
('Scufita Rosie', 'sapca rosie', 500),
('Alba ca zapada', 'alba', 700),('Dexter', 'geniu',
8),('Scooby', 'vorbeste si are pete', 20);
```

Rezultat – conţinutul tabelului „personaje” din baza de date „test”

nume	puteri	varsta
Scufita Rosie	sapca rosie	500
Alba ca zapada	alba	700
Dexter	geniu	8
Scooby	vorbeste si are pete	20

Sursa fişierului *selector.php*

```
<html>
<head>
<title>Datele personajelor</title>
</head>

<body>
<?php
// ne conectam la serverul specificat
//  in contul si parola specificata
$conexiune=mysql_connect("localhost","root","");
mysql_select_db("test",$conexiune); // selectam baza de date
$cerere="SELECT * from personaje"; // pregatim cererea
$raspuns=mysql_query($cerere); // executam cererea

// extragem liniile din raspuns pana nu mai sunt linii
while($arrayul=mysql_fetch_array($raspuns))
{
     echo $arrayul["nume"] . " are puterile <strong>" .
$arrayul["puteri"] . "</strong> si are <strong>" .
// afisez o linie de rezultat
$arrayul["varsta"] . "</strong> ani.<br>";
}
?>
</body>
</html>
```

Rezultat

Scufita Rosie are puterile **sapca rosie** si are **500** ani.
Alba ca zapada are puterile **alba** si are **700** ani.
Dexter are puterile **geniu** si are **8** ani.
Scooby are puterile **vorbeste si are pete** si are **20** ani.

Sursa fişierului *formular.html*

```
<html>
<head>
<title>Pagina de inserare in baza de date</title>
</head>
<body bgcolor="white">
<form name="form1" method="post" action="inserator.php">
  <table border="1">
    <tr>
      <td><div align="center">Nume</div></td>
      <td><div align="center">Puteri</div></td>
      <td><div align="center">Varsta</div></td>
    </tr>
    <tr>
      <td><input type="text" name="Nume"></td>
      <td><input type="text" name="Puteri"></td>
      <td><input type="text" name="Varsta"></td>
    </tr>
    <tr>
      <td colspan="3"><div align="center">
          <input type="submit" name="Trimite" value="Trimite">
        </div></td>
    </tr>
  </table>
</form>
</body>
</html>
```

Rezultatul

Nume	Puteri	Varsta
Pacala	smecher	100
	Trimite	

Sursa fişierului *inserator.php*

```
<?php
$conexiune=mysql_connect("localhost","root","");
mysql_select_db("test",$conexiune);
$cerere="INSERT INTO personaje values
('$_POST[Nume]','$_POST[Puteri]','$_POST[Varsta]')";
mysql_query($cerere);
echo "Datele au fost inserate.";
?>
```

Rezultatul

Datele au fost inserate.

Rezultatul în tabelul „personaje”

nume	puteri	varsta
Scufita Rosie	sapca rosie	500
Alba ca zapada	alba	700
Dexter	geniu	8
Scooby	vorbeste si are pete	20
Pacala	smecher	100

Rezultatul după rularea *selector.php*

Scufita Rosie are puterile **sapca rosie** si are **500** ani.
Alba ca zapada are puterile **alba** si are **700** ani.
Dexter are puterile **geniu** si are **8** ani.
Scooby are puterile **vorbeste si are pete** si are **20** ani.
Pacala are puterile **smecher** si are **100** ani.

În exemplul de mai sus, fişierul „personaje.sql" conţine comenzile SQL pentru crearea şi încărcarea tabelului „personaje" cu date. Scriptul „selector.php" extrage datele din acest tabel şi generează o pagină dinamică cu aceste date. Fişierul „formular.html" oferă o interfaţă grafică pentru inserarea în tabelul „personaje", prin intermediul unui formular care trimite datele scriptului „inserator.php" pe server, care, la rândul său, inserează în tabel datele primite din formular.

Exemplu:

```
<?php
$leg = mysql_connect('localhost','root','') or
die(mysql_error());
$ok = mysql_select_db(nume_baza_de_date,$leg);
?>
```

În cazul în care nu se poate realiza conexiunea sau accesul la baza de date, se vor afişa mesaje de eroare.

Alte funcţii, cunoscute sub numele de funcţii mysql, care oferă facilităţi în crearea interogărilor, transmiterea lor către server şi manipularea rezultatelor sunt:

Funcţia	Efect şi exemplificare
mysql_affected_rows (identificatorul conexiunii stabilite prin mysql_connect())	Întoarce drept rezultat numărul de linii afectate de interogarea anterioară şi +1 în caz de eroare. Dacă nu este furnizat identificatorul conexiunii, se presupune implicit valoarea ultimei conexiunii realizate.
mysql_client_encoding (identificatorul conexiunii stabilite prin mysql_connect())	Rezultatul este numele setului de caractere.
mysql_pconnect (nume_server, nume_utilizator_daca_exista,parola_daca_exista, tip)	Deschide o conexiune persistentă cu server-ul MySQL. Tipul poate fi una dintre constantele: **MYSQL_CLIENT_SSL**, **MYSQL_CLIENT_COMPRESS**, **MYSQL_CLIENT_IGNORE_SPACE** **MYSQL_CLIENT_INTERACTIVE**.
mysql_close()	Închide o conexiune MySql.
mysql_query (sirul de caractere asociat interogarii[, identificatorul	Trimite o interogare spre server pe baza conexiunii stabilite anterior.

legaturii cu baza de date])	

mysql_result(variabila în care s-a păstrat rezultatul unei interogari, al câtelea element din linia unei interogări se doreşte a fi accesat **)**	Preia datele din rezultatul interogării.
	
	`<?php`
	`$leg = mysql_connect('localhost', 'user', 'pass');`
	`$rez = mysql_query('SELECT titlu FROM colectie.carti');`
	`echo mysql_result($rez, 2); // selectează numele celei de+a treia cărţi`
	`mysql_close($link);`
	`?>`

mysql_create_db(numele bazei de date noi[, legatura])	Creează o bază de date mysql .
mysql_data_seek(rezultatul interogarii, nr. randului din rezultat)	Mută poziţia pointerului intern al bazei de date. `<?php` `$leg = mysql_connect('localhost', 'root', '');` `$bd = mysql_select_db('colectie');` `$interog = 'SELECT cod,titlu FROM carti';` `$rez = mysql_query($interog);` `}` `for ($i = mysql_num_rows($rez) - 1; $i >= 0; $i=$i-1) {` `echo $row[$i] . ' ' . $row['cod'];` `}` `mysql_free_result($rez);?>`
mysql_db_query	Trimite o interogare mysql.
mysql_drop_db	Şterge o bază de date mysql.
mysql_fetch_array (variabila asociată rezultatului interogarii[, tipul rezultatului]**)**	O linie obţinută în urma interogării este asociată cu un array(tablou). Tipul rezultatului poate fi: **`MYSQL_ASSOC`**, **`MYSQL_NUM`** şi ca valoare implicită **`MYSQL_BOTH`**.

mysql_fetch_lengths(variabila asociată rezultatului interogării**)**	Obţine lungimea fiecărei componente a rezultatului.
	variabila asociată rezultatului interogarii
	
	`<?php`
	`$leg = mysql_connect('localhost', 'root', '');`
	`$bd = mysql_select_db('colectie');`
	`$rez = mysql_query("SELECT title, autor FROM carti WHERE cod = '1200'");`
	`$rand = mysql_fetch_assoc($rez);`
	`$n = mysql_fetch_lengths($rez);`

	echo
	print_r($rand);
	print_r($n);
	?>

mysql_fetch_row	Fiecare linie din rezultat este vazută ca un sir de enumerări.
mysql_list_dbs	Bazele de date disponibile pe server MYSQL .
mysql_list_tables	Listează numele tabelelor dintr-o bază de date MySQL.

mysql_num_fields(variabila asociată rezultatului interogării, un întreg reprezentând al câtelea câmp să fie evaluat **)**	Redă numărul de câmpuri(coloane) din rezultat unei interogări.
	<?php
	$rez = mysql_query("SELECT cod, titlu FROM carti WHERE cod = '1200'");
	if (!$rez) {
	echo mysql_error();
	}
	$n = mysql_field_len($rez, 1);
	echo $n;?>

mysql_num_rows	Redă numărul de linii din rezultat .
mysql_tablename	Redă numele tabelului din care face parte câmpul specificat.
mysql_field_len	Redă lungimea câmpului specificat ca parametru.
mysql_field_name	Numele câmpului specificat din rezultat.
mysql_field_seek	Setează pointerul rezultatului pe offset-ul unei înregistrări specificate.
mysql_field_table	Redă numele tabelului în care se află câmpul.
mysql_field_type	Redă tipul unui câmp specificat al rezultatului.

La obţinerea de informaţii legate de client, host(gazdă), server sau protocolul utilizat se folosesc cu success funcţiile:

mysql_get_client_info – informaţii legate de client;

mysql_get_host_info – informaţii legate de host;

mysql_get_proto_info – informaţii legate de protocolul folosit;

mysql_get_server_info – informaţii legate de server ;

mysql_info – informaţii despre cea mai recentă interogare;

Pentru preluarea mesajelor de eroare predefinite se utilizează funcţiile:

mysql_errno – valaorea numerică a mesajului de eroare al ultimei operaţiuni MySQL

mysql_error – texul mesajului de eroare al ultimei operaţiuni MySQL .

Exemplu:

Se consideră că există creată o bază de date “colecţie” din care face parte tabelul “carti” cu următoarea structură:

	cod	int(25)
	titlu	varchar(35)

	autor	varchar(100)
	aparitie	tinyint(4)
	recenzie	varchar(150)

Se creează fişierul formular.htm:

```
<html>
<head>
<title>formular </title>
<meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1">
</head>

<body>
<form action="php_mysql.php" method="post" name="form1">
<font color="#0000FF">Introduceti un cod de la care sa inceapa
cautarea:</font>
<br>
<input type="text" name="codul">
<input name="buton_ok" type="submit" value="ok">
</form>
</body>
</html>
```

şi fişierul php_mysql.php:

```
<html>
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1">
</head>
<?
$codul=(int)$_POST['codul'];
$leg = mysql_connect('localhost','root','') or
die(mysql_error());; //pentru afisarea unui mesaj de eroare
$ok = mysql_select_db("colectie",$leg);
$interogare="select titlu,autor from carti where cod
>=$codul";
$rezultatob = mysql_query($interogare);
$rowob=mysql_fetch_row($rezultatob);

?>
<body>

 <table id="simplu" bordercolor="#0000FF" bgcolor="#99FFFF">
  <tr><td><b>Codul introdus a fost:  </b> </td><td><?phpecho "
$codul";?></td></tr>
 <?phpfor ($i=1; $i<= mysql_num_rows($rezultatob);$i=$i+1)

     {$a=mysql_field_name($rezultatob,0);

$b=mysql_field_name($rezultatob,1);
                                                    echo
"<tr><td>$a</td><td>$rowob[0]</td></tr>
```

```
<tr><td>$b</td><td>$rowob[1]</td></tr>
                            ";
                                                        $rowob =
mysql_fetch_array($rezultatob, MYSQL_NUM);//se selecteaya
urmatoarea linie din rezultat
                                                        }

   ?>
</table>
<h3>Alte informatii legate de interogare:</h3>
<table>
<?php
$a=mysql_get_client_info();
$b=mysql_get_host_info();
$c=mysql_get_proto_info ();
$d=mysql_get_server_info();
echo "<tr><td>informatii legate de client </td><td>
$a</td></tr>;
       <tr><td>  informatii legate de host</td><td>$b
</td></tr>
       <tr><td> informatii legate de protocolul  folosit
</td><td>$c</td></tr>
       <tr><td>informatii legate de server
</td><td>$d</td></tr>";

$bd= mysql_list_dbs($leg);
$n=mysql_num_rows($bd);$i=0;

while ($i<$n)
{
print "<tr><td>". mysql_db_name($bd, $i) ."</td></tr>";
$i=$i+1;
}

?>

</table>
</body>
</html>
```

9.15. Lucrul cu fişiere

PHP permite procesarea fişierelor primite de la client sau chiar pe harddiscul serverului pe care rulează scriptul, deşi acest lucru este contraindicat, din motive de securitate.

Exemplu „jurnal”:

Sursa fişierului *jurnal.php*

```
<?php
$fisu=fopen("log.txt","a");
fwrite($fisu,"Conectare cu ".$_SERVER["HTTP_USER_AGENT"]." de
la ".$_SERVER["REMOTE_ADDR"]." la data de ".date("l dS of F Y
h:i:s A").".\n");
```

```
echo "Conectare cu
<strong>".$_SERVER["HTTP_USER_AGENT"]."</strong> de la
<strong>".$_SERVER["REMOTE_ADDR"]."</strong> la data de
<strong>".date("l dS F Y h:i:s A")."</strong>.";
?>
```

Rezultatul în browser

Conectare cu **Opera/9.10 (Windows NT 5.1; U; en)** de la **127.0.0.1** la data de **Saturday 23rd September 2006 12:01:45 AM**.

Rezultatul în fişierul *log.txt*

Conectare cu Opera/9.10 (Windows NT 5.1; U; en) de la 127.0.0.1 la data de Saturday 23rd September 2006 12:01:45 AM.

Exemplul de mai sus generează o pagină de răspuns despre tipul browserului, adresa clientului, data şi ora conectării. Aceleaşi informaţii sunt salvate şi în fişierul „log.txt”, cu diferenţa că fişierul nu este suprascris, se adaugă câte o linie nouă pentru fiecare acces; de aceea, scriptul scrie câte un „\n”, adică linie nouă în fisierul text, la sfârşitul fiecărei linii. Acest fişier nu este de tip html, ci txt, deci linia nu trebuie terminată cu „
”. Fişierul „log.txt” se poate vizualiza cu orice editor de text.

9.16. Mesaje de eroare

Diferitele funcţii de acces la baze de date, pentru includerea unor fişiere, crearea unor obiecte sau multe altele sunt pasibile de rezultate nedorite sau de imposibilitatea executării lor. Din acest motiv, există funcţii pentru preluarea mesajelor de eroare în diferite situaţii. Iată doar câteva dintre ele:

Funcţia	Efectul	Exemplul

msql_error()	Returnează un şir de caractere care reprezintă mesajul de eroare asociat ultimei funcţii mysql.	`<?php`
		`$leg = msql_connect("localhost","","" )`
		`or die("imposibilitate de conectare" . msql_error());`
		`msql_select_db("colectie", $leg)`
		`or die("imposibil de accesat baza de date" . msql_error());`
		`?>`

error_log(mesajul[,tipul mesajului[, destinatia[, alti parametri]])	Transmite un mesaj de eroare spre un server, un port TCP sau unui fişier.	`<?php`
		`error_log("mesajul", 1, " cont@ mail_server_SMTP.domeniu ","Subject: exemplu From:mine@mailserver.domeniu \n");`
		`?>`

error_reporting([un întreg care specifică nivelul de eroare])	Specifică care erori vor putea fi accesate prin cod. Returnează un întreg.	`<?php`
		`error_reporting(E_ERROR | E_WARNING);// va reda doar cateva erori din execuţia codului`
		`?>`
		Observaţie: E_ERROR va impune oprirea

		execuţiei codului iar E_WARNING va genera o avertizare însă fără ca execuţia să fie oprită.

9.17. Exerciţii PHP

1. Executaţi exemplele existente în cadrul capitolului.
2. Creaţi un formular html prin care sa introduceti 2 numere şi cu ajutorul unui script php să calculaţi suma, diferenta, înmulţirea şi împărţirea celor două numere.
3. Aflaţi toate informaţiile posibile despre server.
4. Comparaţi două siruri introduce print-un formular de către utilizator şi tipăriţi-le în ordine crescătoare.
5. Introduceti un cuvânt de la tastatură si eliminaţi toate spatiile albe. Verificati dacă şirul este vid şi ăn caz afirmativ, solicitaţi reintroducerea unui alt şir de caractere.
6. Stergeţi fişierul exa.php şi încărcaţi în browser fişierul exb.php. Ce se întâmplă? Înlocuiţi require cu include. Reîncărcaţi.
7. Încercaţi să identificaţi utilizatorul care accesează o anumită pagină html sau php prin setarea unei variabile cookie.
8. Încercaţi să identificaţi şi alte nivele de eroare.
9. Creaţi o bază de date cu numele “personal” şi un tabel în cadrul acestei baze de date, “date_personale” şi “salarii”. Câmpul comun pentru cele 2 tabele va fi cnp.

   Creaţi un formular prin care să introduceţi în tabelul “date_personale” numele, prenumele, tatăl, mama, cnp, serie si număr de buletin, data naşterii, adresa, casatorit sau nu, copii daca există. Introduceţi cel puţin 5 înregistrări.
10. Selectaţi din tabelul date_personale, barbaţii născuţi după 1970. Calculaţi numărul de înregistrări obţinute şi tipăriţi valoarea pe ecran.
11. Introduceţi în tabelul “salarii”, cu ajutorul unui formular şi a unui script php: salariul net, salariul brut, cuantumul impozitului, data ultimei majorări, motivul pentru majorare, data de incepere şi de încheiere, dacă este cazul, a contractului de muncă. Verificaţi dacă au fost realizate correct inserările în tabelă. Veţi introduce prin formular şi codul numeric personal sau numele celui pentru care vor păstrate asociat informaţii.