

CHIP

Special

Pe CD
Peste 60
de scripturi
PHP

Creați ușor un

Website dinamic

- ▶ Interactivitate pentru pagina dumneavoastră
- ▶ Secretele profesioniștilor la îndemâna oricui

PHP și MySQL
pe înțelesul tuturor

Securitate

Cele mai bune
**TIPS &
TRICKS**

Soluția completă pentru un
magazin online

Apache 1.3.27 și 2.0.44, PHP 4.3.1, MySQL 3.23 pentru Windows și Linux; **Browsere:** Opera 7.03, Mozilla 1.2.1; **Scripturi complete:** PhpBB 2.0.4, phpMyAdmin, phpAdsNew, Gallery, PHPOpenChat, MyPhpMoney; **Editoare PHP și HTML:** PhpEdit, Programmer Studio, Quanta, EditPlus, conText; **MySQL GUI:** MySQLFront 2.5, SQLyog, JSoft Security Manager for MySQL, myFrontEnd; **Acceleratoare PHP:** Turck MMCache for PHP, PHP Accelerator; **Clienti FTP:** FlashFXP, BulletProof FTP

LEVEL DOT RO

pe gustul fiecărui



www.level.ro **forum** ☺**știri** ☺**trivia** ☺**faza zilei**

review ☺**hardware** ☺**lifestyle** ☺**fanathic**



Oana Săulescu
oana@chip.ro

Mai repede, mai ușor

În ziua în care mi s-a dat de făcut un site de prezentare de produse, site care la un calcul preliminar urma să conțină nu mai puțin de 500 de pagini și peste 2000 de poze, am știut că trebuie să existe o alternativă pentru o astfel de muncă sistemică. La urma urmelor, mi-am zis, doar Yahoo și Amazon nu au angajat mii de oameni pentru a face zilnic modificările necesare pe paginile lor, nu? Ca să adaugi sau să modifici o pagină a unui site trebuie să descarci de pe server, să aplici modificările, să găsești și celelalte pagini care ar avea de suferit de pe urma schimbării linkurilor și referințelor și să aplici și la acestea modificările de rigoare, iar apoi să urci totul înapoi pe server... Perspectiva unui site cu 500 de pagini actualizat în acest mod, în fiecare zi a săptămânii m-a făcut să văd negru în fața ochilor chiar dinainte de a mă apuca de lucru. Ulterior am aflat că beneficiarul site-ului pe care urma să îl fac dorea să și poată vinde produsele on-line, să țină legătura cu cumpărătorii prin intermediul site-ului și să aibă o evidență a vânzărilor, stocurilor și clienților site-ului actualizată la cerere. Știam că dacă ar fi fost să fac site-ul HTML, beneficiarului i-ar fi trebuit un întreg departament pentru a putea face față cerințelor.

Am aflat de bazele de date și limbajele server-side întrebând în stânga și în dreapta care ar fi fost soluția pentru problema mea. O idee vagă despre ce înseamnă o baza de date aveam deja, știam că având o serie de informații similare le pot organiza în format tabelar, ca într-o foaie de lucru Excel, pentru a putea găsi, sortă și prelucra datele mai rapid. Dar nu știam cum se poate lega o bază de date de o pagina web, altfel decât salvarea în format HTML, soluție care mi se părea inacceptabilă, aş fi ajuns înapoi de unde plecasem, la download, modificat HTML, upload. Am întrebat din nou și din multitudinea de răspunsuri enigmatice ca Perl, ASP sau

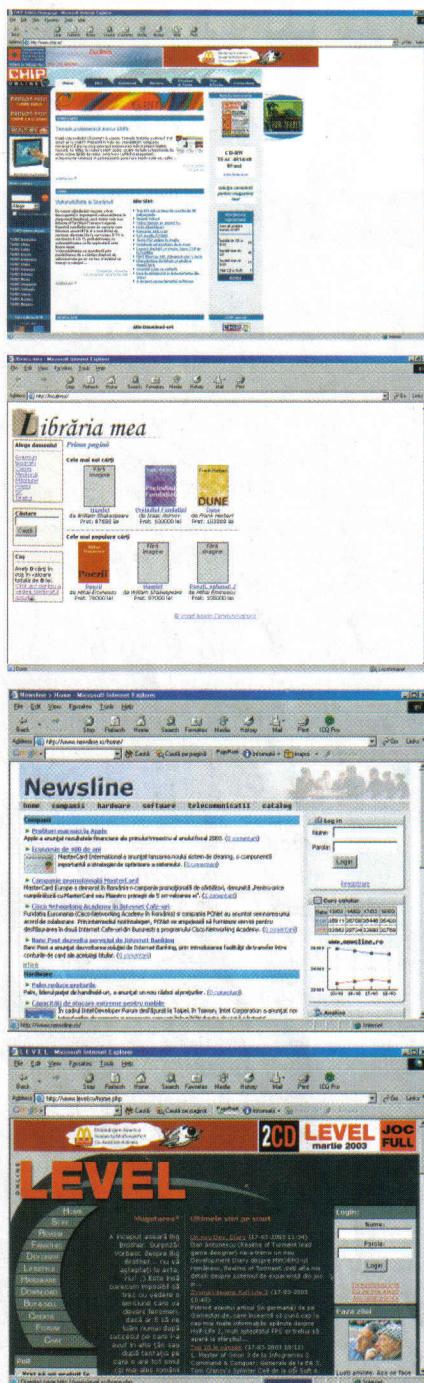
Java, unul se ieva cel mai des: PHP.

PHP este un limbaj server-side care poate, printre altele, să interacționeze cu o baza de date pentru a afișa sau a modifica informațiile din ea. Server-side înseamnă că rulează și face cereri către baza de date la server care la rândul lui „servește” paginile HTML rezultate. Astfel, clientul nu are nevoie de pluginuri, programe adiționale sau setări speciale în browser pentru a vedea rezultatul, pagina HTML creată cu ajutorul PHP.

Acest Special se adresează celor care au ajuns într-un impas asemănător, care vor să învețe mai mult sau pur și simplu să și facă viață de web designer mai ușoară.

Trebuie să aveți cunoștințe minime de HTML și acces la un server cu PHP și MySQL (acesta poate fi și calculatorul personal pe care să instalați cele necesare de pe CD-ul alăturat urmând pas cu pas indicațiile prezентate în capitolul Instalare). Nu este necesar să aveți cunoștințe preliminare de baze de date sau programare. Singura cerință este să exersați și să aplicați chiar voi toate exemplele prezентate aici. De multe ori, un rezultat concret vă poate lămuri mai bine decât un paragraf descriptiv.

Crearea de site-uri folosind PHP și baze de date este unul din cele mai bune lucruri pe care le-ați putea învăța. Societatea evoluează, Internetul la fel și, conform ultimei statistici văzute de mine, pe piața locurilor de muncă, HTML-ul și în cădere liberă în timp ce limbajele server-side sunt din ce în ce mai cerute. PHP poate fi doar începutul pentru voi, noțiunile învățate în acest CHIP Special le puteți considera ca un punct de plecare în învățarea chiar a limbajelor „serioase” precum Java sau C++. Cea mai mare realizare va fi însă eliminarea rutinei și micșorarea timpului de lucru pe care îl veți acorda oricărui proiect, oricără de complex sau de simplu.



CUPRINS

3 Editorial

6 Introducere

Dezvoltarea Internetului este în al treilea stadiu de dezvoltare, iar „dinamic” și „interactiv” sunt atrăgătoarele esențiale ale oricărui site de succes.- PHP și MySQL sunt uneltele ideale pentru un site dinamic.

7 Cei trei magnifici

Sfaturi pentru o instalare rapidă a unelțelor necesare pentru construcția unui site folosind.

9 Cu creionul pe hârtie

Stabilirea obiectivelor.

11 SQL-ul meu și-al tău

Baza de date este coloana vertebrală a unui site dinamic. În acest capitol veți învăța tot ce aveți nevoie să știți pentru a construi și a lucra cu o bază de date.

20 Programare pentru toți

Introducere în limbajul PHP.

Interactivitate folosind PHP în combinație cu formularele HTML.

Învățați cum să folosiți PHP și MySQL pentru a induce dinamism în paginile dumneavoastră.

36 Noul site pas cu pas

Creați primele pagini dinamice și interactive ale site-ului.

51 Stăpânul datelor

Secțiunea de administrare a site-ului.

65 Tips & Tricks

Cum să vă protejați de neplăceri.

Impressum

Redacția poate fi contactată la:

Telefon: 0268-415158, 418728, 0723-570511, 0744-754983;

Fax: 0268-418728; **E-mail:** redactie@chip.ro

Adresa redacției: 2200 - Brașov, str. N.D. Cocea nr.12

Adresa pentru corespondență:

2200 - Brașov, Oficiul Poștal 2, Căsuța Poștală 4

Director General: Dan Bădescu (dan_bădescu@vogelburda.ro)

Director tehnic: Daniel Dănilă-Békési (dan_danila@vogelburda.ro)

Redactor: Oana Săulescu (oana@chip.ro)

Secretar de redacție: Cristiana Vrăbioiu (cristiana_vrăbioiu@chip.ro)

Coperta: Virgil Popa (virgil@chip.ro)

Grafică, DTP: Virgil Popa (virgil@chip.ro)

Contabilitate și administrație: Maria Parge, Eva Szaszka, Adrian Dumitru (contabilitate@vogelburda.ro)

Reclamă: Zsolt Bodola (zsolt_bodola@vogelburda.ro), Cristian Pop (cristian_pop@vogelburda.ro)

Marketing: Leonte Mărginean (leonte_mărginean@vogelburda.ro), Oana Leu (oana_leu@vogelburda.ro)

Distribuție: Ioana Bădescu (ioana_bădescu@vogelburda.ro), Ioan Soiu (iancu_soiu@vogelburda.ro)

Pregătire filme: Artúr Repro LTD.

Montaj și tipar: Veszprémi Nyomda RT, Ungaria

Relații internaționale: [http://www\(chip.ro/html/about/international.php3](http://www(chip.ro/html/about/international.php3)

Editura: Vogel Burda Communications SRL

Sediul editurii: 2200-Brașov, Str. N.D. Cocea nr.12

Copyright:

În România: Vogel Burda Communications SRL Brașov

În Germania: Vogel Burda Holding GmbH, Würzburg, Director: Hermann Paul

ISSN 1453-7079

Manuscrisle, inclusiv în format electronic, expediate redacției devin proprietatea editurii. Editura își rezervă dreptul de modificare a materialelor primite, precum și a datei de apariție. Reproducerea integrală sau parțială a articolelor, informațiilor sau a imaginilor apărute în revistă este permisă numai cu acordul scris al editurii. Redacția nu își asumă răspunderea pentru greșeli și inadverențe apărute în materialele colaboratorilor și ale inserenților.

Soluția completă...

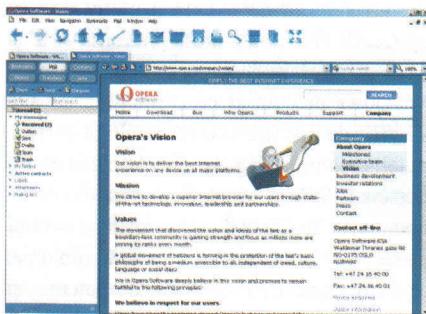
Sub acest moto am reunit pe CD o colecție a celor mai bune aplicații pentru crearea unui site web dinamic, în mare parte absolut gratuite.

Uneltele de bază

Bineînțeles, crearea unui site performant, aici înțelegând unul dinamic, precum și punerea lui în funcțiune, nu poate fi făcută fără anumite programe, dedicate. Din acest motiv, pe CD am inclus serverul web Apache, sistemul de baze de date MySQL și PHP.

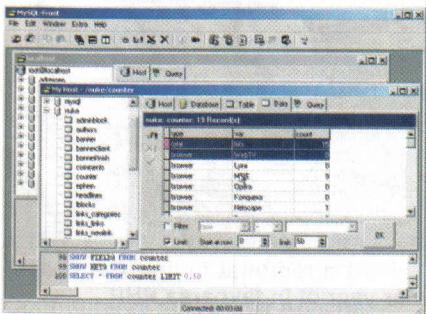
Browsere

Pentru testarea paginilor pe care le veți face este clar nevoie de un browser. Acesta a fost și unul din motivele pentru care am inclus pe CD ultimele versiuni a două browsere consacrate: Mozilla și Opera.



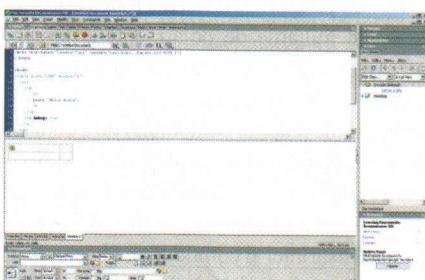
Extras

Ce ar însemna crearea/programarea unui site web fără ajutorul micilor programe care salvează timp? O muncă de luni de zile. De aceea, pe CD veți găsi numeroase astfel de utilitare: clienți vizuali pentru conectarea și lucrul cu baza de date, editoare PHP, drivere MySQL pentru conectarea din .Net, Java și Python.



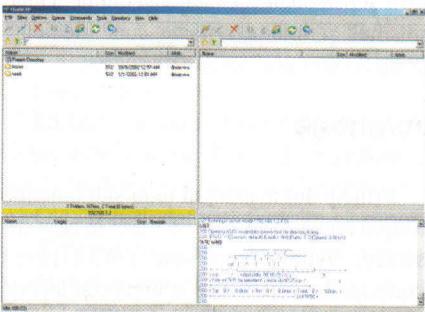
Macromedia

Nu este designer web care să nu fi auzit de firma Macromedia și produsele sale. Acestea ușurează foarte mult munca de creare, programare și finalizare a unui site web, datorită unor unelte ca Homesite, Flash MX, Dreamweaver, ca să amintim numai câteva din versiunile de probă incluse pe CD-ul atașat revistei. Folosind Homesite 5 sau Dreamweaver MX, crearea de site-uri web dinamice va fi mai ușoară deoarece amândouă oferă tag-tips și autocomplete pentru HTML și PHP.



Clienți FTP

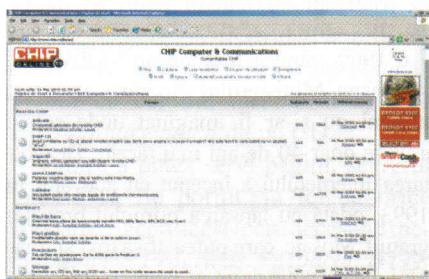
După realizarea și testarea site-ului vine și momentul în care acesta trebuie transferat pe un server conectat la Internet. Pe CD veți găsi cei mai populari clienți de FTP cu care veți putea realiza transferul rapid și ușor.



Scripturi complete

Am inclus și o serie de scripturi complete, gratuite, pe care le puteți utiliza pentru a vă îmbogăți site-ul. Printre cele mai importante se numără PhpBB 2.0.4, un forum pe care trebuie doar să îl configurați pentru a închega o comunitate pe site-ul dvs., phpMyAdmin, o

unealtă cu ajutorul căreia vă puteți conecta la baza de date de oriunde ati fi, Gallery, cu care vă puteți pune poze online, phpMyChat, un sistem de chat web-based, precum și multe altele.



Ajutor în caz de nevoie

Pe CD pe lângă Apache 1.3.27 și 2.0.44, PHP 4.3.1 și MySQL 3.23 pentru Windows și Linux veți găsi și manualele PHP și MySQL în format HTML și CHM (Windows Help).

CD-ul inclus poate fi utilizat în conformitate cu parametrii definiți în standardul Philips - Yellow Book. Editura nu-și asumă responsabilitatea asupra eventualelor pagube provocate de utilizarea CD-ului în alti parametri decât cei stabiliți în standardul menționat anterior.

NOTĂ

Interfața CD-ului CHIP Special este concepută să ruleze optim pe o placă grafică ce suportă minim o rezoluție de 800 x 600 și o adâncime a culorii de 16 biți. De aceea, nu este recomandată folosirea acesteia într-un mediu ce nu oferă minimul necesar! Interfața poate fi rulată atât sub Windows 95/98/Me, cât și sub Windows NT/2000/XP.

Din cauza multitudinii de configurații, redacția CHIP Special nu își poate asuma nici o responsabilitate în eventualitatea în care apar probleme în funcționarea interfeței și a aplicațiilor. Programele care au intrat în componența CD-ului CHIP Special au fost testate și selectate cu grijă în redacția CHIP Special. Totuși, redacția nu își poate asuma nici o responsabilitate pentru funcționarea anormală a software-ului și nici nu poate fi făcută responsabilă pentru eventualele daune produse. CD-ul CHIP Special a fost verificat împotriva virusilor cu următoarele programe antivirus (în ordine alfabetică): BitDefender Professional 6.4 (furnizat de Softwin), F-Secure AntiVirus 5.40 (furnizat de Infodesign), Kaspersky Lab Anti-Virus 4.0 (furnizat de Kaspersky Lab - Rusia) și RAV 8.5 (furnizat de GeCAD).

Pentru orice întrebări legate de aplicațiile de pe CD, vă rugăm să contactați telefonic, prin fax sau prin e-mail, autorii programelor respective.

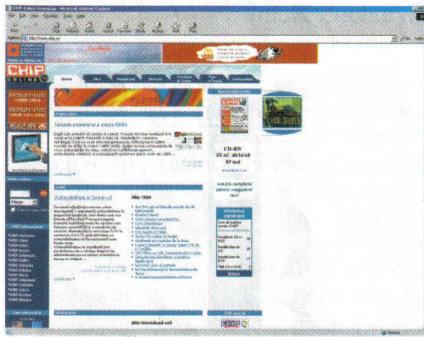
ATENȚIE! Pentru rularea corectă a interfeței CD-ului vă recomandăm setarea unei rezoluții minime de 800 x 600, o adâncime a culorii de 16 biți și folosirea opțiunii Small Fonts!

Introducere

Încotro ne îndreptăm?

Dezvoltarea Internetului este în al treilea stadiu de dezvoltare, iar „dinamic” și „interactiv” sunt atributele esențiale ale oricărui site de succes.

In 1960, când a fost lansat ARPANET, rețeaua Ministerului Apărării al SUA, nimeni nu și-ar fi imaginat unde se va ajunge doar 40 de ani mai târziu. Dezvoltarea Internetului a început cu adevărat în 1993, odată cu lansarea primului browser gratuit, Mosaic, care putea afișa text și imagini. În mai puțin de un an, numărul utilizatorilor a ajuns la 2 milioane. David Siegel, autorul cărții „Creating killer websites” se referă la site-urile create în această perioadă ca „Site-uri de primă generație”, lineare, limitate ca posibilități dar îndeajuns de funcționale pentru transmiterea informației.



Articolele prezente pe amândouă paginile au fost introduse o singură dată și s-au aranjat „sigure”: cele mai noi la început, cele mai vechi la sfârșit.

A doua generație a început odată cu specificațiile pentru HTML 1 și competiția Netscape-Microsoft pentru dominarea pieței. Orientarea în această perioadă a fost înspre „vizual” ca mod de comunicare.

În prezent ne aflăm în a treia generație, a dinamismului și interactivității. Lumea progresează în pas rapid iar web designerii trebuie să țină pasul, în această meserie evoluția este cuvânt cheie.

Numărul internauților crește de la o zi la alta, la fel ca și așteptările lor: ei doresc informație proaspătă și posibilitatea de a interacționa. Paginile statice nu mai sunt de ajuns pentru a asigura fidelizarea utilizatorilor. În plus, în ziua de astăzi oricine poate face un site cu ajutorul aplicațiilor vizuale precum Macromedia DreamWeaver sau Microsoft Frontpage. Pentru a sta în fața concurenței, un web designer trebuie să ofere mai mult și mai rapid decât ceilalți.

Într-o statistică făcută pe ejobs.com, cerearea de programatori HTML este în cadrul liberă în timp ce limbajele server-side sunt cele mai cerute. Ca webdesigneri, ori ne adaptăm noilor cerinței ori îngroșăm coada la Oficiul forțelor de muncă.

Se cere...

În prezent aproape jumătate de miliard din populația globală are acces la Internet. În 2002 39,7% din utilizatorii de Internet au făcut cumpărături cu o valoare totală estimată la 47,98 miliarde de dolari. Site-urile care oferă produse sau servicii sunt într-o continuă creștere. Utilizatorii tend să prefere cumpărăturile online datorită comodității de a comanda un produs fără efort sau deplasare. Aproximativ 98% din e-cumpărători spun că intenționează să cumpere online și în viitor datorită ușurinței și comodității. Un studiu al Cyberatlas relevă faptul că vânzările online au cel mai mare potențial de dezvoltare din întreg sectorul economic. Site-urile firmelor nu mai sunt doar informative ci oferă utilizatorului posibilitatea de a comanda produse sau servicii. La fel și site-urile care oferă știri: ele nu mai sunt demult doar simple pagini de ziar în format electronic, ci oferă utilizatorului posibilitatea de a interacționa, de a-și personaliza experiența online. Dinamismul și interactivitatea sunt elemente obligatorii pentru un site de succes.

Avantaje

Principalul avantaj al unui site dinamic asupra unuia static este ușurința în administrare. Adăugarea sau modificarea conținutului este floare la ureche și atât de simplu încât puteți delega pe oricine să se ocupe de această sarcină. Deoarece conținutul este stocat în baza de date sau în fișiere text, cei care se ocupă de conținut nu au nevoie să știe vreun pic de HTML sau organizarea sau layoutul paginilor.

De asemenea, dacă trebuie să schimbați logo-ul pe fiecare pagină a site-ului este de ajuns să faceți modificarea într-un singur fișier în loc să luați fiecare pagină în parte, să o modificați și să o uploadați înapoi pe server.

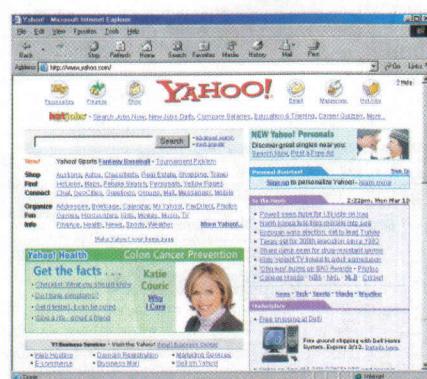
Într-un site dinamic se pot adăuga elemente de interactivitate foarte ușor. Metodele puse la dispoziție de aplicațiile server-side ne pot ajuta să transformăm un simplu site într-o comunitate activă.

Tendința este că în viitor, toate electronicele din jurul nostru să fie conectate la Internet. Deja putem accesa site-uri wap prin intermediul telefonului mobil și să inventați până și frigidere care face singur comenzi online. De aici se pot ridica o serie de probleme de compatibilitate. Un site creat pentru web va trebui recreat pentru a putea fi accesibil și de pe telefonul mobil sau ultimul rănet de mp3 player cu conectare la Internet. Având conținut separat de structură sau de prezentare, într-o bază de date, se poate crea foarte ușor un duplicat al site-ului original, cu un format potrivit destinației.

De ce PHP și MySQL?

Cel mai puternic motiv pentru folosirea PHP și MySQL ca unele pentru crearea unui site dinamic este faptul că amândouă sunt open-source și oricine le poate utiliza fără costuri suplimentare. Un alt motiv este că aplicațiile create sunt cross-platform: PHP și MySQL rulează atât pe sisteme Linux și Windows precum și pe o mulțime de alte sisteme de operare mai rar întâlnite. Astfel, puteți crea un întreg website pe calculatorul personal, fără să instalați un nou sistem de operare, iar la sfârșit doar să transferați site-ul pe serverul Linux și să ruleze fără probleme.

În prezent, PHP este atât de puternic și versatil încât până și liderul portalurilor, Yahoo!, a renunțat la limbajul său proprietar, yscript, în favoarea PHP. Aceasta nu înseamnă că PHP este complicat sau dificil: dimpotrivă, PHP și MySQL sunt foarte ușor de învățat, chiar și pentru cei nefamiliarizați cu programarea sau bazele de date. Acest Special dorește să demonstreze acest lucru - tot ce va trebui să faceți este să citiți și să exersați chiar voi toate exemplile prezentate în continuare.



Yahoo! a renunțat la limbajul proprietar yscript în favoarea PHP

Instalare și cerințe de sistem

Cei trei magnifici

Instalarea celor trei componente necesare funcționării unui server web poate da bătăi de cap chiar și unui utilizator experimentat în ale calculatoarelor. Omiterea adăugării unei linii într-un fișier de configurare vă poate scoate pe albi încercând să aflați de ce lucrurile nu funcționează aşa cum ar trebui.

Dar de ce e atât de complicat, ați putea întreba? Ei bine, pentru că explica acest lucru, trebuie întâi să vă povestesc ce se petrece de fapt pe serverul dumneavoastră când scrieți în bara de adrese a browserului <http://www.chip.ro/1.php>, spre exemplu. Când apăsați Enter, browserul trimite către serverul de HTTP <http://www.chip.ro> o cerere: „afișeză-mi pagina 1.php a site-ului www.chip.ro”. Acest server este Apache. Apache știe să servească (de aceea se și numește server) browserului doar pagini HTML. Paginile HTML pot fi deja pe server sau create, tot acolo, de către alte aplicații, la cererea serverului de web și servite ca HTML browserului clientului. În momentul în care cerem pagina test.html de pe server, Apache o servește imediat.

Dacă în schimb cerem o pagină cu altă extensie, php în cazul de față, Apache cauță să vadă dacă este configurat să servească pagini cu această extensie și ce program se ocupă de ele. Astfel, atunci când cerem o pagină cu extensia php, serverul Apache va trimite mai departe cererea către PHP, „se cere un fișier PHP, ocupă-te de această cerere și dă-mi înapoi o pagină pe care să o afișez clientului”. PHP preia cererea, rulează codul din program și dacă vede că se cer informații din baza de date MySQL, o accesează, extrage informațiile cerute și construiește o pagină HTML pe care o va trimite serverului Apache pentru ca acesta să o servească clientului.

Un exemplu concret: prima pagină a site-

ului chip.ro este un fișier PHP care ia din baza de date cele mai noi știri adăugate de redactori și le trimită înapoi către Apache care le afișează vizitorului. Dacă vă uitați în sursa paginii nu veți vedea decât cod HTML. PHP rulează „în umbra”, pe server (de aceea se și numește limbaj „server-side”) și conlucră cu Apache astfel încât acesta din urmă să poată afișa clientului un rezultat interpretabil de către browser - în acest caz o pagină HTML. Apache nu este predefinit să lucreze cu PHP și viceversa. În vederea colaborării între cele două va trebui ca după instalarea lor să modificați fișierele de configurare.

În continuare vă voi ghida prin instalarea și configurarea Apache, PHP și MySQL.

Apache

Pentru un site aveți în primul rând nevoie de un server HTTP. Alegerea noastră s-a îndreptat spre Apache datorită flexibilității sale, portabilității, siguranței și extensibilității. Pe CD-ul alăturat puteți găsi două versiuni Apache, pentru cele mai populare sisteme de operare. Vă voi ghida în continuare prin procesul de instalare pe un sistem Windows aşa cum majoritatea dintre dumneavoastră aveți.

Ultima versiune Apache pentru Windows este 2.0.44. Kitul de instalare îl găsiți în directorul *apache 2.0.44\binary* pe CD. Totuși, datorită faptului că suportul PHP pentru Apache 2 este încă experimental, sfatul meu este să instalați Apache 1.3.27 pe care îl găsiți pe CD în directorul *apache 1.3.27\binary*. Vă va fi mult mai ușor să instalați această versiune dar folosiți-o doar pentru uzul personal, pentru a putea face pagini dinamice fără să fiți nevoiți să vă cumpărați hosting la un ISP sau să pierdeți timp cu configurații exhaustive. Nu folosiți serverul Apache pe Windows 95 sau 98 pentru a vă pune online site-urile, ci doar pentru a le crea în liniște în vederea publicării lor ulterior.

pe un server securizat.

În timpul instalării Apache, în momentul în care vi se vor cere informații despre server, folosiți următoarele date:

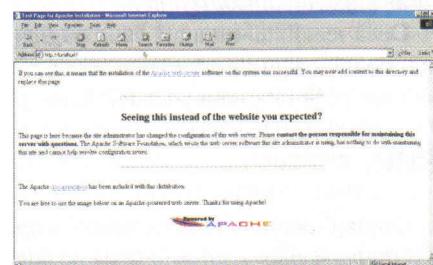
Network domain: localhost

Server name: localhost

Administrator's Email Address: adresa dvs. de mail.

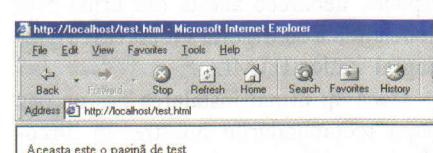
Notă: Dacă instalați Apache 2, va trebui ca după încheierea instalării să suprascrieți manual fișierul libapr.dll din *c:\Program Files\Apache Group\Apache 2\bin* cu cel din directorul *apache 2.0.44\binary\patch* de pe CD.

Pentru a putea servi pagini, serverul trebuie să fie pornit, la fel ca orice aplicație. Faceți acest lucru accesând meniul *Start > Programs > Apache HTTP Server > Start Apache in console* care va porni programul într-o consola. Pentru a testa, deschideți un browser și accesați adresa <http://localhost>. Dacă pagina afișată arată ca în imaginea alăturată, serverul este pornit și funcționează.

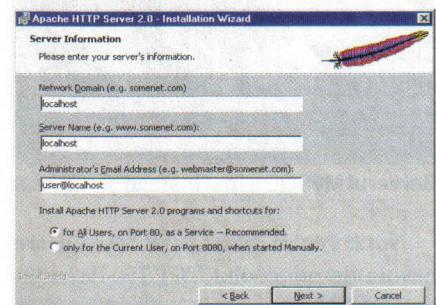


Prima pagină servită de server după instalare.

Pentru a nu fi nevoiți ca de fiecare dată când vreți să aveti serverul pornit să rulați programul manual, puteți să îl porniți ca serviciu și el va porni automat odată cu Windows. În Windows NT, 2000 și XP, odată cu instalarea programului, este instalat și serviciul Apache. Îl puteți seta să pornească automat accesând Services (Windows Service Control Manager) din Control Panel (Administrative Tools în Windows 2000). În Windows 95/98 există suport pentru servicii însă este doar experimental. Pentru ca Apache să pornească automat la fiecare repornire a calculatorului, trebuie să îl instalați întâi ca atare. Folosiți o consolă MS-DOS prompt pentru a intra în directorul *C:\Program Files\Apache Group\Apache* și a tasta *apache -i -n „apache”*. Dacă mesajul primit este „The



O pagină HTML scrisă de utilizator și servită de Apache.



Configurarea serverului Apache.

apache service has been installed successfully." puteți să și porniți imediat serviciul (fără a reboota) scriind în consolă:

```
apache -n „apache” -k start.
```

Dacă vă hotărăți să nu mai folosiți serviciul Apache, tastați în command prompt apache -u -n „apache” și serviciul va fi dezinstalat, iar programul nu va mai fi pornit automat odata cu Windows.

Document root

Document root este directorul în care află fișierele HTML și PHP ce vor fi servite de server. În cazul de față el este directorul C:\Program Files\Apache Group\Apache\htdocs. Orice fișier punetă în document root va fi disponibil la adresa <http://localhost/numefisier>. Puteți testa acest lucru creând o pagină test.html în care să scrieți:

```
<p>Aceasta este o pagină de test</p>
```

Copiați fișierul test.html în document root și tastați în bara de adrese a browserului <http://localhost/test.html>.

Dacă rezultatul nu arată ca în pagina anterioară, asigurați-vă că nu există un alt server HTTP pornit (IIS, PWS, etc.).

PHP

Copiați conținutul directorului php 4.3.0\binary\php-4.3.0-Win32 de pe CD în c:\php, iar apoi fișierul C:\php\php 4ts.dll în c:\windows\system.

În continuare, pentru a vă ușura instalarea puteți copia fișierul php.ini din directorul chip.config de pe CD în c:\windows.

Alternativ, dacă doriți să faceți configurarea manuală, deschideți fișierul c:\php\php.ini-dist cu Notepad, căutați textul doc_root= și schimbați-l cu doc_root="C:\Program Files\Apache Group\Apache\htdocs". Căutați apoi linia extension_dir = și schimbați-o cu extension_dir=„C:\php”. Salvați fișierul, redenumiți-l php.ini și copiați-l în directorul Windows.

Apache+PHP

Încă nu vom putea accesa PHP din Apache deoarece acesta din urmă este deocamdată configurat să serveasca doar pagini statice HTML.

Dacă ati urmat instalarea pas cu pas după recomandările noastre, în directorul chip.config de pe CD veți găsi fișierul de configurare pentru Apache, httpd.conf, gata pregătit, fiind suficient

să suprascrieți cu acesta corespondentul său din directorul C:\Program Files\Apache Group\Apache\conf. Dacă doriți să faceți configurarea manuală (în cazul în care ati instalat PHP în alt director decât cel indicat aici), deschideți cu Notepad fișierul C:\Program Files\Apache Group\Apache\conf\httpd.conf și adăugați următoarele 3 linii:

```
LoadModule php4_module c:/php/sapi/
php4apache.dll
AddModule mod_php4.c
AddType application/x-httdp-php .php
```

Căutați apoi linia DirectoryIndex index.html și modificați-o în DirectoryIndex index.html index.php. Pentru ca schimbările să fie recunoscute va trebui să reporniți Apache.

Să testăm dacă totul funcționează cum trebuie. Creați cu Notepad un document text în care scrieți:

test.php

```
<?
print "Acesta este un test. Apache+PHP
funcționează";
?>
```

Salvați-l cu numele test.php și accesați URL-ul <http://localhost/test.php>.



Primul script PHP!

Atenție! Dacă Windows-ul dumneavoastră este setat să ascundă extensiile pentru fișierele recunoscute, documentul creat cu Notepad va fi de fapt numit test.php.txt și adresa <http://localhost/test.php> nu va putea fi accesată deoarece test.php nu există. În cazul acesta aveți două soluții: ori continuați să lucrăți cu Notepad, dar va trebui să dezactivați ascunderea extensiilor pentru fișierele cunoscute, ori folosiți un editor PHP din cele puse la dispoziție pe CD.

MySQL

Pentru a instala ultima versiune MySQL pentru Windows nu este nevoie decât să porniți setup.exe din directorul mysql 3.23\binary. Deoarece MySQL este un server, ca și Apache, va trebui întâi să-l porniți dacă dorîți să lucrăți cu el. Pentru a-l porni, rulați C:\mysql\bin\mysqld.exe pe

Dacă nu merge...

Note pentru instalarea și rularea Apache Nu funcționează?

Dacă atunci când accesați adresa <http://localhost> nu vedeați o pagină cu mesajul "If you can see this, it means that the installation of the Apache web server software on this system was successful.", verificați întâi dacă serverul este pornit și dacă nu cumva aveți alt server HTTP (PWS, IIS) care rulează.

Windows 95

Dacă aveți Windows 95 trebuie să downloadați și să instalați Windows Socket 2 Update de la Microsoft înainte de a începe instalarea Apache. Windows NT 4.0, 98, ME și 2000 nu au nevoie de acest update. Adresa de download este: http://www.microsoft.com/windows95/downloads/contents/wuadmintools/s_wunet_workingtools/w95sockets2/default.asp

Windows XP

Pentru a putea folosi Apache pe sistemele Windows XP trebuie să aveți Windows XP Service Pack 1 instalat. Îl puteți obține de la <http://www.microsoft.com/windowsxp/pro/downloads/servicepacks/sp1/default.asp>

Atenție!

Pe unele sisteme poate fi necesar să copiați fișierele php4apache.dll și php4ts.dll din directorul /php/ în directorul /windows.

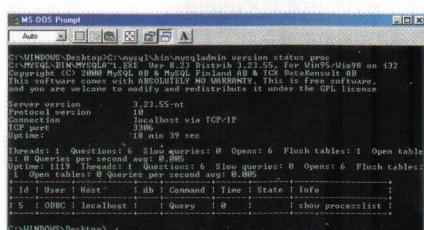
Windows 95/98 sau C:\mysql\bin\mysqld-nt.exe pe NT/2000/XP.

Ca serverul de MySQL să pornească automat la fiecare repornire a Windows, puteți adăuga un shortcut în Start Menu/Start up către C:\mysql\bin\mysqld.exe (respectiv mysqld-nt.exe). Este recomandat ca pe Windows NT/2000/XP să setați serviciul MySQL să pornească automat.

Să testăm: după ce ati pornit serverul mysqld scrieți următoarea linie în command prompt:

```
C:\mysql\bin\mysqladmin version status proc
```

Dacă output-ul este asemănător celui din fereastra alăturată, serverul MySQL funcționează.



Serverul MySQL a fost instalat și rulează.

Notă: Pentru ca să ruleze, MySQL are nevoie de protocolul TCP/IP. Pe sistemele cu Windows NT4 trebuie să aveți instalat Service Pack 3.

Organizarea preliminară

Cu creionul pe hârtie

Regula numărul 1 în programare este organizarea preliminară. De acesta depinde succesul acțiunii la care ne vom înhâma. Pentru a scrie cod bun trebuie să știm destul de clar care ne sunt obiectivele înainte de a scrie chiar și un tag HTML. Câteva minute de organizare preliminară ne pot scuti de revenirea la cod pentru a aduce modificări sau chiar rescrierea întregii aplicații.

Organizarea preliminară este o practică obișnuită în firmele de software și ar trebui să fie legea de căpătăi a oricărui programator web. Dacă știm dinainte cum vrem să funcționeze o aplicație, când trecem la scrisul codului com poate să ne concentram exclusiv asupra programării fără să avem alte preocupări. De aceea vom lua un creion și o hârtie și vom descrie fiecare pagină a site-ului, funcționalitățile ei și modul în care utilizatorul va interacționa cu elementele acesteia.

Să definim întâi ce dorim să obținem de la site-ul nostru. Dorim ca vizitatorii să cumpere cărți, la fel ca într-un magazin „real”. Ei vor putea să se plimbe prin spațiul virtual, să vadă copertele cărților, să răsfoiască două-trei pagini, să își pună fiecare carte într-un coș de cumpărături și la sfârșit să se ducă la casă pentru a plăti. Fiecare carte va fi însoțită și de o scurtă descriere deoarece nu vom avea un librărie care să facă acest lucru pentru noi. La fel ca la librărie, cărțile vor fi aranjate după domeniul căruia aparțin

pentru a le putea localiza mai ușor. Și tot pentru că nu avem un librărie căruia să îi cerem direct „Dați-mi seria Fundației de Asimov” sau „Ce cărți aveți despre PHP?”, vom pune la dispoziția vizitatorilor noștri un motor de căutare pentru a găsi rapid ceea ce îi interesează fără a fi nevoiți să piardă timp prețios căutând prin rafturi. Deoarece un magazin virtual aduce foarte mult cu unul real, o să vă prezint în continuare fiecare pagină a site-ului prin analogie cu o librărie reală.

Vitrina

Ne aflăm la ușa librăriei. Putem vedea rafturile de cărți marcate sugestiv „Manuale”, „Medicina”, „Poezii”, „Aventuri”, „SF”, „Calculatoare”, etc. În vitrină se află câteva cărți alese pe sprâncenă de către librărie: cărțile foarte populare și noile apariții.

La fel va fi și prima pagină a magazinului nostru virtual: vom prezenta genurile de cărți pe care le avem disponibile, best-seller-urile, precum și cele mai noi intrări. Pentru că spațiul virtual ne permite, tot de la intrare vom oferi vizitatorului și un coș de cumpărături precum și opțiunea de căutare pentru a găsi o carte rapid, la fel cum ne ducem într-o librărie reală direct la primul angajat să îl întrebăm atunci când căutăm o anume carte.

Pe raft

Intrăm în magazin pentru a cumpăra o carte pentru un prieten a căruia zi de naștere este duminică. Acestua îi plac povestirile SF așa că ne îndreptăm spre sirul de rafturi deasupra căruia scrie „SF” pentru a vedea ce am putea alege. Acest lucru este echivalent cu a da click pe unul din domeniile prezentate pe prima pagină și a intra în pagina dedicată aceluia domeniu unde se

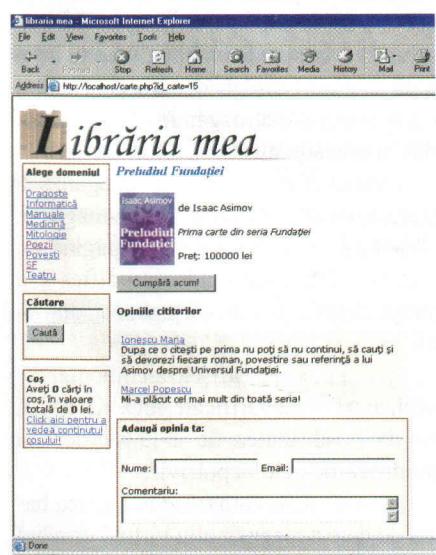
află înșirute cărțile aparținând acestuia. Pe această nouă pagină îi vom arăta vizitatorului cărțile ordonate după autor și titlu însoțite fiecare de o scurtă descriere și de prețul cărții. Titlul cărții va fi sub formă de link astfel încât vizitatorul să poată apăsa pe el și să vadă mai multe detalii despre cartea respectivă.

Cartea

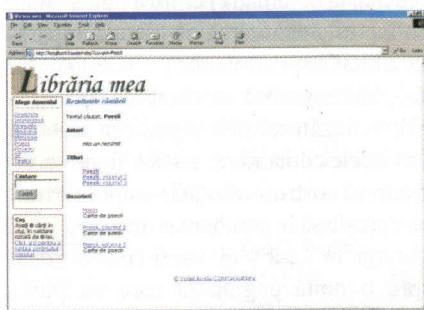
În magazinul real putem lua o carte din raft pentru a-i vedea coperta și a o răsfoi. Cam același lucru vom face și în site-ul nostru. Apăsând pe link-ul reprezentat de titlul cărții se deschide o nouă pagină în care vom prezenta detaliiile cărții, imaginea copertei, prețul și butonul de pentru adăugarea cărții în coș. Vom mai adăuga și o secțiune în care să putem vedea comentariile celor ce au citit deja cartea, precum și posibilitatea de a adăuga un nou comentariu. Aceasta funcție este deosebit de utilă atunci când cumpărăm o carte pentru cineva și domeniul ne este străin. Având la dispoziție comentariile celorlalți putem alege o povestire SF bună pentru prietenul nostru chiar dacă n-am citit-o niciodată.

Coșul de cumpărături

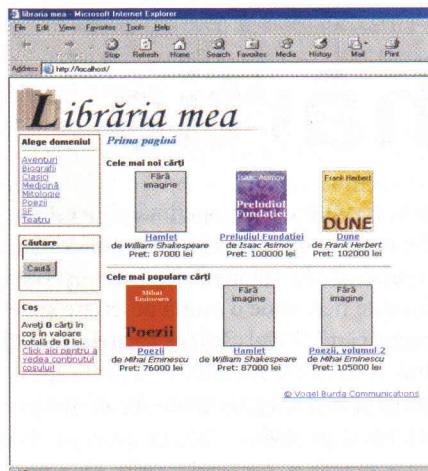
Atunci când un vizitator va apăsa butonul „Adaugă în coș” el va fi redirecționat spre o nouă pagină în care îi este afișat conținutul coșului și două opțiuni: să continue cumpărăturile sau să meargă la casă pentru a plăti. În cazul în care dorește să mai cumpere, aplicația noastră va ține minte alegerea vizitatorului pentru a-i o putea afișa la o nouă privire în coș. Valoarea însumată a cărților adăugate în coș o vom afișa permanent



Detaliile cărții.



Lista rezultatelor căutării.



Prima pagină: cele mai noi și mai populare cărți.

Într-un colț al paginii, pentru a îmbunătăți experiența vizitatorilor pe site-ul nostru (aceștia nu vor mai trebui să calculeze permanent în minte starea bugetului de care dispun). Utilizatorul va putea apăsa în orice moment pe coș pentru a-i revizui conținutul (cărțile alese până în momentul respectiv). Dacă totalul depășește bugetul alocat el poate scoate o carte-două din coș. Dacă în schimb își aducem aminte în ultimul moment că un alt prieten căuta aceeași carte ca cea pe care dorește să o dăruiască, utilizatorul poate modifica numărul de bucăți din coș pentru a cumpăra două în loc de una.

Căutare

Să presupunem că vizitatorul va veni pe site-ul nostru căutând o anume carte. Pentru această situație vom avea funcția de căutare. El va putea căuta după mai multe criterii: autor, numele cărții sau descriere. Rezultatul îi va fi afișat la fel ca în pagina de domeniu: titlu, autor și descriere, cu titlul sub formă de link către pagina dedicată cărții.

Casa

Deoarece în România încă nu se pot face tranzacții online adevărate, soluția cea mai la îndemâna pentru a putea vinde online este trimiterea de către cumpărător a banilor prin mandat poștal sau ordin de plată, urmând ca după confirmarea plății vânzătorul să îi expedieze acestuia produsele comandate. Astfel, după ce vizitatorul nostru s-a hotărât asupra cărților ce dorește să le cumpere și apasă butonul „Mergi la casă”, el va fi redirectionat spre o nouă pagină în care va putea introduce adresa unde dorește să le primească. După introducerea acestor de-

talii va apăsa pe „Încheie tranzacția”, îi va fi afișată o pagină de mulțumire și dumneavaoastră veți fi anunțat prin email de cerere pentru a o putea onora cât mai rapid.

Administrare

Pe lângă acestea vom avea nevoie și de o secțiune de administrare cu ajutorul căreia să adăugăm sau să scoatem cărți din rafturile magazinului virtual, să definim domenii și să revizuim comentariile vizitatorilor (există și posibilitatea ca vreun glumeț să adauge un comentariu nepotrivit care ar trebui șters sau modificat). Tot în secțiunea de administrare vom avea o pagină unde să vedem ultimele cereri și să bifăm tranzacțiile efectuate cu succes după primirea banilor și trimiterea cărților cerute către destinatar.

Structura site

După ce am terminat cu cerințele, vom trece la împărțirea pe pagini. Astfel avem:

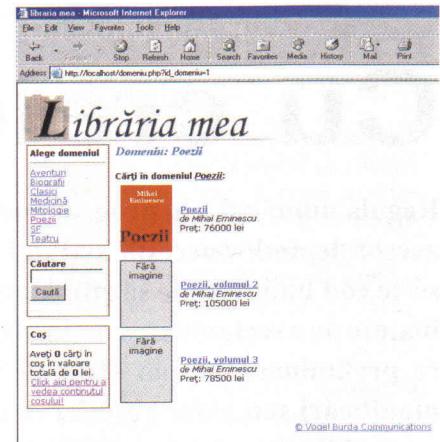
Zona publică

`index.php` – prima pagină
`domeniu.php` – pagina dedicată unui domeniu (SF, Aventuri, etc.) în care afișăm cărțile din domeniul respectiv

`carte.php` – pagina în care putem vedea detalii despre carte, coperta, comentariile utilizatorilor și în care fiecare utilizator va putea adăuga un comentariu personal

`adauga_comentariu.php` – va procesa textul trimis de utilizator și îl va adăuga în baza de date

`cos.php` – pagina unde vizitatorul poate vedea și edita conținutul coșului de



Lista cărților din domeniul Poezii.

cumpărături

`casa.php` – formularul de introducere a datelor pentru expediere și plată

`prelucrare.php` – pagina care va prelucra datele trimise prin formularul din `casa.php`: le introduce în baza de date, vă trimite un email de notificare și apoi afișează „Multumim pentru că ați cumpărat de la noi”.

Zona de administrare

Toate paginile din această zonă vor fi protejate de parolă pentru ca numai dumneavaoastră să aveți acces și se vor afla în directorul **admin**:

`index.php` – afișează un formular de login;

`login.php` – verifică informațiile din formular și autorizează sau nu accesul mai departe, la pagina următoare;

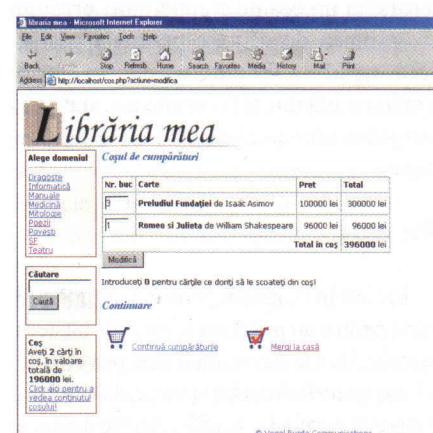
`comenzi.php` – cu ajutorul acestei pagini vom face cele mai dese acțiuni. Ea va conține comenzi încă neonorate și vă va da posibilitatea de a le marca. În momentul în care primiți banii pentru o comandă o puteți marca „Vândut” și aceasta nu va mai apărea în listă. Dacă în schimb o comandă este mai veche de o lună o puteți șterge deoarece probabil cumpărătorul s-a răzgândit și nu s-a mai dus la poștă pentru a vă trimite banii;

`adaugare.php` – pentru adăugare sau ștergere de cărți și domenii din e-magazin;

`modificare_stergere.php` – pagina cu ajutorul căreia vom putea modifica sau șterge date de pe site: cărți, autori, domenii

`opinii.php` – afișează comentariile utilizatorilor neverificate încă de noi și ne dă posibilitatea de a le șterge sau modifica pe cele nepotrivite.

În continuare vom trece la crearea bazei de date a magazinului virtual urmând ca apoi să facem paginile PHP din care va fi compus site-ul.



Coșul de cumpărături.

Introducere în MySQL

SQL-ul meu și-al tău

Baza de date este coloana vertebrală a unui site dinamic. În acest capitol vom afla tot ce avem nevoie să știm pentru a construi și a lucra cu o bază de date.

La fel ca într-o librărie, cărțile din magazinul nostru virtual vor trebui aranjate după domeniul căruia îi aparțin, nu vom pune laolaltă într-o harabură fără sens romane de aventuri cu manuale școlare și cărți de psihologie aprofundată.

În aplicația noastră va trebui să organizăm datele cărților în aşa fel încât să le putem găsi ușor după domeniu, să leordonăm după titlu sau să le alegem pe cele aparținând unui anumit autor. Soluția cea mai flexibilă care să corespundă acestor cerințe este baza de date.

Colecții de date

Baza de date este, în trei cuvinte, o colecție organizată. Ca exemple v-aș putea da cartea de telefon sau programul TV din jurnalul pe care îl citi dimineața la cafea. În cartea de telefon găsim numele și prenumele abonatului, adresa și numărul de telefon, toate acestea ordonate după nume, pentru a căuta mai ușor o anumită persoană. Dacă am avea cartea de telefon în format electronic într-o bază de date am putea face o ordonare după numărul de telefon în ordine crescătoare pentru a afla cine are cel mai mic număr. Am putea de asemenea să aflăm căți abonați au nume care încep cu T, căți dintre ei au prenumele Ion, numărul de abonați din oraș, pe care stradă sunt cele mai multe posturi telefonicе și aşa mai departe.

În programul TV informațiile au următoarea structură: canalul, ora difuzării, numele programului. Acestea sunt de obicei ordonate după canal și ora difuzării, astfel:

Canalul A

16:00	Desene animate: Balanel și Miaunel
18:00	Film: Tacerea mieilor
19:00	Stiri
20:00	Film: Matrix
22:00	Muzica

Canalul B

16:00	Muzica
17:00	Documentar: Al doilea razboi mondial

18:00 Stiri

19:00	Desene animate: Mica sirena
20:00	Talkshow financiar
21:00	Film serial: StarTrek (ep. 140)
16:00	Program pentru copii

Canalul C

17:00	Desene animate: Dexter
18:00	Film serial VIP (ep. 101)
19:00	Stiri
20:00	Film: Lord of the Rings
22:00	Documentar: Marele alb

Să ne imaginăm că tăiem cu foarfeca din ziar fiecare nume de program și ora de difuzare.

Atunci ne-am putea ordona programele după ora difuzării astfel încât să putem distinge mai ușor ce ne interesează într-un anume interval orar:

Ora	Nume program	Canal
16:00	Desene animate: Balanel și Miaunel	Canalul A
	Muzica	
	Program pentru copii	Canalul C
	Documentar: Al doilea razboi mondial	Canalul B
17:00	Desene animate: Dexter	Canalul C
	Film: Tacerea mieilor	Canalul A
	Stiri	Canalul B
	Film serial VIP (ep. 101)	Canalul C
19:00	Stiri	Canalul A
	Desene animate: Lilo și Stitch	Canalul B
	Stiri	Canalul C
	Film: Matrix	Canalul A
20:00	Talkshow financiar	Canalul B
	Film: Lord of the Rings	Canalul C
	Film serial: StarTrek (ep. 140)	Canalul B
21:00	Muzica	Canalul A
	Documentar: Marele alb	Canalul C

Având o bază de date scăpăm și de ziar și de foarfecă deoarece ne putem ordona programele după plac, putem alege să vizualizăm doar programele între ora 17 și ora 22, putem afișa pe ecranul calculatorului doar filmele din program și aşa mai departe.

În măruntaiele bazei de date

Am spus că baza de date este o colecție organizată, acum este momentul să vă arăt și cum este organizată. O bază de date este compusă din unul sau mai multe tabele (*tables*) care, la rândul lor, sunt formate din înregistrări dispuse în câmpuri (*fields*).

Dacă ati lucrat vreodată cu un program de calcul tabelar vă puteți imagina baza de date ca fiind documentul ce conține mai multe foi de lucru (*tablele*), câmpurile ca fiind coloanele tabelelor și o înregistrare oarecare ca fiind un rând din tabel. Să luăm exemplul cărții de telefon. Baza de date va conține un singur tabel cu 4 câmpuri: nume, prenume, adresă, numar de telefon.

Nume	Prenume	Adresa	Numar de telefon
Andu	Paul	Lunii 29	4567893
Baciu	Bianca	Viforului 15	3456874
Cretu	Mihai	Muncii 4	2154686
Fulg	Ramona	Republicii 14	1445328
Zinc	Ghe.	Bd. Garii 2A	6522554

Conținutul fiecărei linii se numește generic înregistrare (*record*). Din prima înregistrare știm că Andu Paul locuiește pe Lunii 29 și are numărul 4567893. Valoarea câmpului *Nume* al primei înregistrări este Andu, iar valoarea câmpului *Prenume* a celei de-a doua înregistrări este Paul. Baza de date a departamentului contabilitate a unui magazin de patetarie va avea mai multe tabele care să conțină tipuri diferite de date. Niciodată nu îi va fi util să aibă furnizorii amestecați cu stocurile și cheltuielile, ci va dori să aibă aceste date separate, dar totodată puse laolaltă în același loc. Ea va avea nevoie de un tabel Stoc care să conțină două câmpuri:

Nume produs	Nr. buc. in stoc
Creioane	3511
Pixuri	5000
Plicuri A4	275

Va mai avea nevoie și de un tabel Cheltuieli care să aibă următoarea structură:

Produs sau serviciu	Cost	Data
Decoratiuni	5.000.000 lei	20.12.2002
Reparatii	700.000 lei	03.01.2003
Jaluzele	12.000.000 lei	04.01.2003

Pe lângă acestea îi vor mai trebui încă câteva tabele pentru salariai, vânzări, etc., pentru ca la sfârșitul anului să centralizeze datele din toate și să facă bilanțul.

Din acest exemplu putem observa că există cazuri în care avem nevoie de mai multe tabele pentru organizarea informației, tabele pe care le ținem într-o singură bază de date pentru o accesare mai ușoară sau pentru a intermedia relații între ele. Dar despre asta mai târziu.

O bază de date poate avea un număr uriaș de tabele și înregistrări. De pe site-ul www.mysql.com aflăm că există la ora actuală baze de date pe MySQL Server cu 60000 de tabele și 5 miliarde de înregistrări. Impresionant, nu? Cum ar fi fost să stocăm această cantitate de informație pe hârtie, în cataloage, și să căutam prin milioane de pagini ce ne interesează?

Ce este MySQL?

MySQL este cel mai popular sistem de management pentru baze de date relaționale deoarece este Open Source adică poate fi folosit fără să fim nevoiți să plătim vreo sumă de bani. MySQL Server a fost creat pentru a lucra cu baze de date mai rapid decât soluțiile existente deja și este folosit de ani buni în medii foarte solicitante.

Într-o bază de date relațională datele sunt stocate în mai multe tabele separate, fiind astfel îmbunătățite viteza și flexibilitatea. Tabelele pot fi legate prin relații definite de noi, fiind astfel posibil să combinăm la cerere datele din mai multe tabele.

Ce înseamnă SQL?

SQL, acronimul pentru „Structured Query Language”, este limbajul standard pentru comunicarea cu bazele de date. Comenzi SQL sunt folosite pentru a interacționa cu baza de date (de exemplu să adauge, să modifice sau să steargă datele).

Alte sisteme de baze de date care folosesc SQL sunt Microsoft SQL Server, Access, Oracle, Sybase, etc.

Baza noastră de date

Pentru a ne putea face o bază de date trebuie ca serverul MySQL să fie pornit. Dacă ați urmat pașii descriși în capitolul instalare, serverul MySQL ar trebui să ruleze deja, el fiind repornit automat la fiecare restartare a calculatorului.

În continuare va trebui să folosim o aplicație cu ajutorul căreia să comunicăm cu serverul. Aplicația se numește simplu mysql.exe și se găsește în același director, c:\mysql\bin\. Vom intra în command prompt apăsând pe Start, apoi pe Run și scriind **command** în câmpul destinat programului ce urmează să fie rulat. Odată aflați în linie de comandă, scriem:

```
c:\mysql\bin\mysql.exe -p -u root
```

iar când ni se cere parola apăsăm ENTER.

Dacă serverul nu este pornit vom primi mesajul de eroare „Can't connect to MySQL server on 'localhost'”.

Serverul de MySQL, ca orice server care se respectă, oferă posibilitatea de a avea mai mulți posesorii de date pe același sistem, fiecare cu drepturile lui. Astfel, serverul unui ISP de exemplu va putea fi folosit de mulți oameni, fiecare având mai multe baze de date și neputând să vadă ce au ceilalți utilizatori ai aceluiași server.

Utilizatorul implicit (și cu drepturi depline, de altfel) este **root**, iar parola inițială este goală. Vom lucra cu aceste date pentru început dar puteți consulta manualul MySQL inclus pe CD pentru a afla cum puteți adăuga utilizatorii noi sau schimba o parolă în MySQL.

SHOW

Odată autentificăți-vă putea comunica cu serverul MySQL folosind comenzi SQL. Trimisarea unei comenzi SQL către serverul de baze de date se mai numește și interogare. Să facem prima noastră interogare și să aflăm dacă există deja vreo bază de date pe server:

```
SHOW DATABASES;
```

```
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| test    |
+-----+
1 row in set (0.00 sec)

mysql>
```

Rezultatul unei interogări SHOW DATABASES.

```
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| test    |
+-----+
2 rows in set (0.00 sec)

mysql>
```

Acetă sunt bazele de date disponibile pe server.

Rezultatul primit de la server va arăta ca în imaginea de mai sus.

Am încheiat interogarea cu punct și virgulă deoarece toate comenziile SQL (cu excepția QUIT și USE) trebuie încheiate astfel pentru a semnala serverului că am terminat de scris propoziția și că poate trece la procesarea cererii. Există cazuri în care o interogare poate fi foarte lungă și atunci ar trebui să o „rupem” în câteva bucăți mai mici, pe mai multe rânduri. MySQL ar da eroare dacă ar observa că propoziția este neterminată aşa că nu uități să încheiați interogările cu punct și virgulă.

Să inspectăm una din bazele de date de pe acest server. Pentru a face acest lucru, spunem serverului cu ce bază de date dorim să interacționăm în continuare:

```
USE test
```

Vom primi mesajul „Database changed” și baza de date ‘test’ va rămâne cea în care ne vom face interogările până la deconectarea de la server sau dacă decidem să folosim o altă bază de date, cu aceeași comandă:

```
USE alta_baza_de_date
```

Setarea unei baze de date ca fiind cea curentă nu ne împiedică totuși să accesăm tabele din alte baze de date în forma *nume_baza_de_date.nume_tabel*. Comanda USE este păstrată pentru compatibilitate cu Sybase.

Tot comanda comanda SHOW o vom folosi pentru a vedea ce tabele există în baza de date test în care ne aflăm:

```
SHOW TABLES;
```

Observăm că baza de date test nu are nici un tabel. Să vedem dacă cealaltă are, schimbând baza de date cu care dorim să interacționăm (USE mysql) și repetând

```
mysql> SHOW TABLES;
+-----+
| Tables_in_test |
+-----+
| func      |
+-----+
1 row in set (0.00 sec)

mysql>
```

Nu este nici un tabel în baza de date test.

```
CREATE TABLE tabel_test2(camp1
TEXT, camp2 INT, camp3 TINYINT);
```

Interogările SQL se pot scrie pe mai multe rânduri cu condiția ca ultima linie să se încheie cu punct și virgulă.

comanda SHOW TABLES.

O altă utilizare a lui SHOW este SHOW COLUMNS care afișează informații despre coloanele unui tabel.

CREATE și DROP

Am spus că vom face o bază de date special pentru site-ul magazinului nostru de cărți. Să trecem atunci la treabă. Vom scrie în linia de comandă:

```
CREATE DATABASE librarie;
```

Serverul ne afișează „Query OK” deci baza de date a fost creată. Va trebui să mai spunem serverului că aceasta este baza de date cu care urmează să interacționăm în continuare:

```
USE librarie
```

Am putea să folosim interogarea SHOW TABLES și aici, însă ea nu ne-ar arăta mai mult decât știm deja: că noua bază de date nu are încă nici un tabel. Să facem, deci, unul:

```
CREATE TABLE tabel_test
(camp_test TEXT);
```

Temă: executați comenzi SHOW TABLES și SHOW COLUMNS pentru noua bază de date și tabelul proaspăt creat.

Să luăm cuvânt cu cuvânt propoziția: CREATE TABLE tabel_test(camp_test TEXT) înseamnă „creează tabelul tabel_test cu un câmp numit camp_text al căruia tip este TEXT. Pentru a afla ce este tipul de date și la ce folosește consultați oglinda „Tipuri de date”.

Să facem încă un tabel, cu trei coloane:

```
CREATE TABLE tabel_test2(camp1
TEXT, camp2 INT, camp3 TINYINT);
```

Câmpurile și tipul de date conținut de acestea din tabelul tabel_test2.

CREATE TABLE tabel_test2(camp1
TEXT, camp2 INT, camp3 TINYINT);

În imaginea alăturată putem vedea mai multe informații despre fiecare câmp. Să le analizăm:

- prima coloană, Field, este numele câmpului;

- a doua, Type, este tipul de date conținut de câmp. TEXT înseamnă că datele de pe coloana respectiva vor fi texte, INT că vor fi numere întregi și TINYINT că în coloana respectivă vor fi introduse doar numere între -128 și 127;

- în a treia dacă este NULL sau nu, a patra, Key arată dacă este index, în a cincea, Default Value, este specificată valoarea implicită și ultima coloană arată ce alte proprietăți mai are câmpul.

Să revenim la magazinul nostru de cărți, să recapitulăm datele de care avem nevoie în site și să încercăm să le organizăm logic. Fiecare carte are un autor, un titlu și o scurtă descriere. Toate sunt de tip text, deci aşa vor fi și tipurile de câmpuri aferente lor:

```
CREATE TABLE carti(
    autor TEXT,
    titlu TEXT,
    descriere TEXT);
```

În cazul în care nu a fost semnalată nici o eroare, tabelul nostru a fost creat. Să verificăm, de dragul exercițiului, acest lucru, cu comenziile SHOW TABLES și SHOW COLUMNS.

Pentru a șterge un tabel, o bază de date, un index sau o coloană dintr-un tabel folosim comanda DROP astfel:

```
DROP TABLE tabel_test;
DROP DATABASE librarie;
```

Deși MySQL are suport pentru diacritice și setul de caractere 8859-2, este preferabil să nu folosiți diacritice în numele bazelor de date, tabelelor sau câmpurilor. De asemenea, nu puteți folosi ca nume de tabel sau de câmp cuvintele rezervate (nume de funcții, tipuri de caractere din MySQL precum create, drop sau column).

Ar mai trebui să știți că puteți folosi nume de tabele care conțin spații dar în practică trebuie să încadrați numele între back-ticks ` (semnul back-tick îl găsiți pe tastă aflată imediat sub Escape):

```
CREATE TABLE `tabel al carui nume
are spatii`(`camp 1`, TEXT);
SHOW COLUMNS FROM `tabel al carui
nume are spatii`;
```

```
SELECT `camp 1` FROM `tabel al
carui nume are spatii`;
```

INSERT

Haideți să introducем câteva date în tabelul carti. Comanda pe care o vom folosi este INSERT și sintaxa este:

```
INSERT INTO tabel(câmp1, câmp2,
câmp3) values(valoare1, valoare2,
valoare3)
```

În românește această comandă s-ar traduce „introdu în câmpul1 valoare1, în câmpul2 valoare2 și în câmpul3 valoare3”, iar în format tabelar ar arăta în felul următor:

câmpul1	câmpul2	câmpul3
valoare1	valoare2	valoare3

Să trecem la treabă și să exersăm comanda INSERT:

```
INSERT INTO carti(autor, titlu,
descriere) VALUES('William Shakespeare',
'Romeo si Julieta', 'Cea mai frumoasă poveste de dragoste a tuturor timpurilor');
```

Am pus ghilimele în jurul fiecărei valori ce urmează a fi introduse pentru a stabili că acela este un text unitar, un string de introdus în câmpul respectiv. Imaginați-vă cum cum ar interpreta serverul comanda noastră dacă am scrie numele autorului în forma *Shakespeare*, *William*. Atunci textul interogării ar fi: INSERT INTO carti(autor, titlu) VALUES(Shakespeare, William, Hamlet) și serverul MySQL ne-ar returna următoarea eroare: “Column count doesn't match value count at row 1” deoarece se așteaptă ca pentru două coloane să aibă de introdus tot două valori separate prin virgulă, nu trei. Dacă am fi specificat că vrem să introducem valori în toate cele trei coloane am fi primit alt mesaj de eroare deoarece MySQL nu ar fi recunoscut textul ‘Shakespeare’ ca fiind string și s-ar fi oprit din execuție.

Dacă dorim, putem să omitem una din coloane ca în exemplul următor unde nu adăugăm nimic în câmpul descriere:

```
INSERT INTO carti(autor, titlu)
VALUES('William Shakespeare',
'Hamlet');
```

Sintaxa lui INSERT poate fi simplificată dacă luăm în calcul toate câmpurile tabelului deoarece le putem omite, menționând doar valorile ce urmează să fie adăugate ca în exemplul următor:

```
INSERT INTO carti VALUES('Mihai Eminescu', 'Poezii', 'Cele mai frumoase poezii ale poetului național');
```

Observăm că nu am mai specificat coloanele în care urmează să introducem date, sintaxa fiind astfel mult simplificată. Dacă dorim să beneficiem de sintaxa prescurtată pentru INSERT dar nu avem o valoare pentru un câmp, putem să nu punem nimic între ghilimelele care delimită valoarea respectivă, ca în exemplele următoare unde nu introducem nici un text (sau un text gol, dacă vreți să îi spuneti aşa) în câmpul descriere:

```
INSERT INTO carti VALUES('Mihai Eminescu', 'Poezii, volumul 2', '');
INSERT INTO carti VALUES('Mihai Eminescu', 'Poezii, volumul 3', '');
INSERT INTO carti VALUES('Alexandru Mitru', 'Legendele Olimpului', '');
INSERT INTO carti VALUES('George Cosbuc', 'Fire de tort - Poezii', '');
INSERT INTO carti VALUES('Mihai Eminescu', 'Geniu pustiu', '');
```

Temă: Adăugați folosind INSERT încă 5 cărți în tabel specificând titlul, autorul și eventual o scurtă descriere.

SELECT

Acum avem baza de date, avem și câteva date introduse... dar cum le vedem? Vom folosi SELECT, cea mai importantă comandă SQL și îi vom vedea câteva din utilizări în exemplele următoare. Am aflat cum putem vedea ce tabele sunt într-o bază de date, hai să aflăm și ce conține un tabel. Să luăm primul tabel din făcut de noi și să afișăm datele din el:

```
SELECT * FROM tabel_test;
```

Ne va fi returnat următorul mesaj:

```
Empty set (0.17 sec)
```

Empty set înseamnă că tabelul nu conține date, e gol. Numărul din paranteze reprezintă secundele care i-au luat serverului ca să returneze un rezultat. Să încer-

Tipuri de date

Să nu amestecăm mere cu pere...

Tipurile de date care apar în coloanele MySQL ar putea părea o complicație inutilă la prima vedere. Oare nu am putea să adăugăm datele în ce formă avem nevoie? Lucrurile nu stau chiar așa. MySQL aloca spațiu pe disc în funcție de tipul de date specificat de utilizator. Dacă în tabelul salariaj aveam o coloană pentru numărul de zile de concediu legal o vom seta ca fiind TINYINT deoarece aceasta este valoarea numerică în care se încadreză (un salariat nu va avea mai mult de 127 de zile de concediu pe an!). Pentru fiecare înregistrare în coloana zile_concediu MySQL va aloca 1 byte de memorie, indiferent dacă un angajat are 2 zile de concediu și altul are 18. Dacă același câmp îl setăm ca INT, MySQL va aloca fiecărui înregistrare în coloană 4 bytes, indiferent de numărul de zile de concediu introduse și astfel pentru fiecare înregistrare (fiecare angajat) se vor pierde 3 bytes. Acesta este doar un exemplu dar puteți avea la un moment dat o bază de date cu 10 milioane de înregistrări și atunci pierderei a 3 bytes la fiecare înregistrare se traduce în câteva zeci de MB de spațiu.

Iată și tipurile de date în bazele de date MySQL, spațiul pe care îl ocupă precum și valorile minime și maxime pe care le pot avea.

Valori numerice:

Tip	Bytes	De la	Până la
TINYINT	1	-128	127
SMALLINT	2	-32768	32767
MEDIUMINT	3	-8388608	8388607
INT	4	-2147483648	2147483647
BIGINT	8	-9223372036854775808	9223372036854775807

Toate tipurile de mai sus au un atribut opțional (nestandard), UNSIGNED. Puteți folosi acest atribut în definirea tipului de date al unei coloane atunci când dorîți să conțină doar valori pozitive. Un câmp de tip TINYINT va putea conține numere între -128 și 127 în timp ce alt câmp, TINYINT UNSIGNED va putea avea valori între 0 și 255. La fel, pentru SMALLINT valorile sunt de la -32768 până la 32767 în timp ce pentru MEDIUMINT UNSIGNED ele pot fi între 0 și 65535.

Dacă într-un câmp TINYINT care are valori între -128 și 127 veți încerca să introduceți o valoare mai mică de -128 ea va fi convertită în cea mai mică valoare admisă, -128. Dacă veți încerca să introduceți o valoare mai mare de 127 ea va fi convertită în cea mai mare valoare admisă de tipul câmpului, 127 în acest caz.

Dată/timp:

Column type	Format
DATETIME	'YYYY-MM-DD hh:mm:ss'
DATE	'YYYY-MM-DD'
TIMESTAMP	'YYYYMMDDhhmmss'
TIME	'hh:mm:ss'
YEAR	'YYYY'

Tipul de câmp TIMESTAMP oferă posibilitatea de a data automat operațiile de tip INSERT și UPDATE. El este compus implicit din 14 caractere pentru formatul 'YYYYMMDDhhmmss' dar putem să specificăm la crearea unui tabel că dorim să conțină mai puține caractere:

Timestamp

Column type	Display format
TIMESTAMP(14)	YYYYMMDDHHMMSS
TIMESTAMP(12)	YYMMDDHHMMSS
TIMESTAMP(10)	YYMMDDHHMM
TIMESTAMP(8)	YYYYMMDD
TIMESTAMP(6)	YYMMDD
TIMESTAMP(4)	YYMM
TIMESTAMP(2)	YY

Stringuri (șiruri)

Tipurile de string-uri în MySQL sunt BLOB, TEXT, CHAR, VARCHAR, ENUM și SET. În tabelul următor puteți vedea mărimea maximă admisă pentru cele mai folosite dintre ele precum și spațiul alocat pe disc pentru fiecare:

Tip	Mărime maximă	Bytes
TINYTEXT sau TINYBLOB	2^8-1	255
TEXT or BLOB	2^16-1 (64K-1)	65535
MEDIUMTEXT or MEDIUMBLOB	2^24-1 (16M-1)	16777215
LONGBLOB	2^32-1 (4G-1)	4294967295

Câmpul de tip BLOB poate conține o cantitate variabilă de informație, similar cu TEXT însă diferă printr-un singur aspect: căutarea într-un câmp BLOB este case sensitive (se face diferență între majuscule și minuscule), iar într-un câmp TEXT nu este.

Tipul VARCHAR este similar tipului TEXT cu deosebirea că într-o coloană de tip VARCHAR putem specifica numărul maxim de caractere admis.

căm cu celălalt tabel, în care știm sigur că am introdus date și să vedem cum arată acestea:

```
SELECT * FROM carti;
```

Acum putem vedea foarte clar structura și conținutul tabelului. Asteriscul * din SELECT * FROM carti este echivalentul lui „tot/toate”, iar comanda s-ar traduce în limba română „arată-mi tot din tabelul test”. Asteriscul îl putem înlocui cu numele unui câmp pentru a vedea doar conținutul acelui câmp. Astfel:

```
SELECT titlu FROM carti;
```

ne va afișa doar conținutul câmpului titlu:

titlu
Hamlet
Hamlet
Poezii
Poezii, volumul 2
Poezii, volumul 3
Legendele Olimpului
Fire de tort - Poezii
Geniu pustiu

Putem selecta să ne fie afișate mai multe coloane, specificate de noi, în orice ordine (nu neapărat în ordinea în care se află în tabel), ca în interogarea următoare unde afișăm coloanele titlu și autor:

```
SELECT titlu, autor FROM carti;
```

titlu	autor
Hamlet	William Shakespeare
Hamlet	William Shakespeare
Poezii	Mihai Eminescu
Poezii, volumul 2	Mihai Eminescu
Poezii, volumul 3	Mihai Eminescu
Legendele Olimpului	Alexandru Mitru
Fire de tort - Poezii	George Coșbuc
Geniu pustiu	Mihai Eminescu

Putem afișa cărțile pentru care câmpul descriere este gol astfel:

```
SELECT * FROM carti WHERE descriere = '';
```

autor	titlu	descriere
William Shakespeare	Hamlet	
M. Eminescu	Poezii, volumul 2	
M. Eminescu	Poezii, volumul 3	
Alexandru Mitru	Legendele Olimpului	
George Coșbuc	Fire de tort - Poezii	
M. Eminescu	Geniu pustiu	

Avem suficiente date pentru a afla câteva utilizări noi și foarte interesante ale comenzi SELECT. Putem adăuga condiții de selectie și putem astfel alege să ne fie afișate doar titlurile cărților de Mihai Eminescu, astfel:

```
SELECT titlu FROM carti WHERE autor='Mihai Eminescu';
```

titlu
Poezii
Poezii, volumul 2
Poezii, volumul 3
Geniu pustiu

Să afișăm toate volumele de poezii din baza de date, indiferent de autor. Dacă executăm interogarea:

```
SELECT * FROM carti WHERE titlu='Poezii'
```

ne va returna doar cartea al cărui titlu este chiar „Poezii”, neluându-le pe celelalte în considerare:

autor	titlu	descriere
Mihai Eminescu	Poezii	Cele mai frumoase poezii ale poetului national

Folosind condiția titlu='Poezii' nu vor fi afișate volumele 2 și 3 din colecția de poezii de Mihai Eminescu și nici cartea intitulată „Fire de tort - Poezii” de Coșbuc. Dacă vrem să obținem toate cărțile care au cuvântul „poezii” în titlu putem face acest lucru înlocuind egalitatea cu operatorul

LIKE și folosind wildcard-uri. Semnul cu funcție de wildcard într-o interogare MySQL este % (semnul * cu care probabil sunteți deja obișnuiți este definit în SQL ca însemnând tot/toate și l-am folosit deja atunci când am utilizat SELECT * FROM carti pentru a afișa toate înregistrările din

(semnul * cu care probabil sunteți deja obișnuiți este definit în SQL ca însemnând tot/toate și l-am folosit deja atunci când am utilizat SELECT * FROM carti pentru a afișa toate înregistrările din

tabel). Iată cum funcționează LIKE și wildcardul %:

```
SELECT autor, titlu FROM carti WHERE titlu LIKE 'poezii%';
```

Vor afișa toate înregistrările în care titlul cărții începe cu cuvântul „poezii”.

autor	titlu
Mihai Eminescu	Poezii
Mihai Eminescu	Poezii, volumul 2
Mihai Eminescu	Poezii, volumul 3

MySQL tratează toate valorile dintr-un câmp de tip TEXT ca fiind case-insensitive astfel încât nu va trebui să țineți cont de literele mari sau mici atunci când vă referiți într-o interogare la valoarea unui câmp. Comanda:

```
SELECT * FROM carti WHERE titlu LIKE '%poezii';
```

va afișa toate înregistrările în care titlul cărții se termină cu cuvântul „poezii”:

volumele *Poezii* de Mihai Eminescu și *Fire de tort - Poezii* de George Coșbuc. Similar,

```
SELECT * FROM carti WHERE titlu LIKE '%poezii%';
```

va afișa toate înregistrările în care titlul cărții conține cuvântul „poezii” oriunde în cadrul textului.

Putem extinde comanda SELECT pentru a adăuga un nou criteriu de selecție.

Dacă dorim să alegem toate cărțile de poezii de Mihai Eminescu fără să ne fie afișat și romanul scris de acesta sau cartea de poezii a lui George Coșbuc, vom trimite către baza de date o interogare cu două criterii: „returnează-mi toate înregistrările unde autorul este *Mihai Eminescu* și titlul cărții conține cuvântul *poezii*”.

```
SELECT * FROM carti WHERE autor='Mihai Eminescu' AND titlu LIKE '%poezii%';
```

autor	titlu	descriere
Mihai Eminescu	Poezii	Cele mai frumoase poezii ale poetului national
Mihai Eminescu	Poezii, volumul 2	
Mihai Eminescu	Poezii, volumul 3	

Dacă vrem să alegem toate cărările care nu sunt scrise de Mihai Eminescu folosim operatorul de inegalitate (semnul !=, opus lui =):

SELECT autor, titlu FROM carti WHERE autor != 'Mihai Eminescu';	
autor	titlu
William Shakespeare	Hamlet
Alexandru Mitru	Legendele Olimpului
George Cosbuc	Fire de tort
Mihai Eminescu	- Poezii

Putem găsi toate titlurile cărările pentru care câmpul descriere este gol astfel:

SELECT titlu FROM carti WHERE descriere != '';	

Analog, folosind operatorul NOT LIKE putem alege toate cărările al căror titlu nu începe cu litera P:

SELECT autor, titlu FROM carti WHERE titlu NOT LIKE 'P%';	
autor	titlu
William Shakespeare	Hamlet
Alexandru Mitru	Legendele Olimpului
George Cosbuc	Fire de tort
Mihai Eminescu	- Poezii

Teme de rezolvat:

- afișați toate titlurile cărările de William Shakespeare;
- afișați toate titlurile care conțin litera T oriunde în textul titlului;
- afișați doar titlurile cărările ale căror autori cu numele care se termină cu litera U;
- afișați numele autorilor care nu se termină cu litera U;

Putem folosi ORDER BY pentru a ordona rezultatele unei selecții, crescător sau descrescător, în funcție de coloana aleasă pentru ordonare.

Iată, spre exemplu, cum putem afișa

autor
Alexandru Mitru
George Cosbuc
Mihai Eminescu
William Shakespeare

autorii din tabel în ordine crescătoare:

```
SELECT autor FROM carti ORDER BY autor ASC;
```

autor
Alexandru Mitru
George Cosbuc
Mihai Eminescu
Mihai Eminescu
Mihai Eminescu
William Shakespeare

Putem avea mai multe criterii de ordonare, ca în exemplul următor unde ordonarea se face după autor ascendent și, pentru fiecare autor, cărările sunt ordonate după titlu descendente:

```
SELECT autor, titlu FROM carti ORDER BY autor ASC, titlu DESC;
```

Putem afla câte înregistrări sunt pentru un criteriu de selecție cu ajutorul lui count(). Putem astfel afla câte înregistrări avem în total în tabel:

```
SELECT count(*) FROM carti;
```

count(*)
8

sau câte înregistrări sunt în tabel al căror câmp ‘autor’ este Mihai Eminescu:

```
SELECT count(*) FROM carti WHERE autor='Mihai Eminescu';
```

count(*)
4

Cu ajutorul instrucției GROUP BY putem „grupa” rezultatele astfel încât să nu vedem duplicatele și să vedem doar valorile unice.

Să vedem, folosind GROUP BY ce autori avem și să ii ordonăm crescător:

```
SELECT autor FROM carti GROUP BY autor ORDER BY autor ASC;
```

Pentru a limita numărul de rezultate returnate, folosim instrucția LIMIT. Dacă avem 10000 de înregistrări și nu dorim să vedem decât primele 3, folosim LIMIT în felul următor:

```
SELECT * FROM carti LIMIT 0,3;
```

Dacă dorim să vedem înregistrările de la 3 la 7:

```
SELECT * FROM carti LIMIT 3,4;
```

iar pentru înregistrările de la 10 la 15:

```
SELECT * FROM carti LIMIT 10,5;
```

Observăm că primul număr din LIMIT este înregistrarea de la care se începe afișarea și al doilea număr este numărul de înregistrări ce urmează a fi returnate.

DELETE

DELETE se folosește pentru ștergerea înregistrărilor dintr-un tabel (ne reamintim că pentru ștergerea tabelelor și a bazelor de date se folosește DROP). Sintaxa lui DELETE este:

```
DELETE FROM table WHERE condiții;
```

Așa cum putem afișa doar înregistrările ce corespund condițiilor noastre, la fel putem condiționa și ștergerea lor în funcție de criteriile specificate.

```
SELECT * FROM carti WHERE autor='Mihai Eminescu';
```

afișează toate cărările de Mihai Eminescu, în timp ce:

```
DELETE FROM carti WHERE autor='Mihai Eminescu';
```

va șterge din tabel toate înregistrările pentru care câmpul autor este Mihai Eminescu, lăsându-le pe celelalte neașteptate.

Observăm două lucruri: asteriscul a dispărut, el fiind inutil. MySQL va șterge un rând întreg la solicitarea dumneavoastră, el nu va putea să steargă (logic) o coloană sau mai multe aparținând unei înregistrări. În al

doilea rând, cu această excepție, sintaxa pentru ștergere seamănă ca două picături de apă cu sintaxa pentru selecție. La fel ca în SELECT putem folosi mai multe condiții și am putea șterge toate înregistrările unde autorul este *Mihai Eminescu* și titlul cărții conține cuvântul *poezii*".

```
DELETE FROM carti WHERE
autor='Mihai Eminescu' AND titlu
LIKE '%poezii%';
```

DELETE FROM carti este echivalentul de ștergere al lui SELECT * FROM carti, adică va șterge toate înregistrările din tabel. Nu îl vom pune în aplicare dar este bine de știut.

UPDATE

Atunci când vrem să modificăm conținutul unei înregistrări nu este nevoie să o ștergem și să o adăugăm în varianta nouă, putem folosi UPDATE care are următoarea sintaxă:

```
UPDATE tabel SET coloana1='noua
valoare a coloanei 1',
coloana2= 'noua valoare a
coloanei 2' WHERE condiții
```

UPDATE modifică conținutul unuia sau mai multor câmpuri în funcție de condițiile specificate. Să modificăm, de exemplu, în coloana autor toate înregistrările Mihai Eminescu și să le înlocuim cu M. Eminescu:

```
UPDATE carti SET autor='M. Eminescu'
WHERE autor='Mihai Eminescu';
```

Execuți SELECT * FROM carti pentru a observa modificarea.

Condițiile pot fi extinse la fel ca în sintaxa SELECT. În următoarea interogare vom modifica câmpul descriere ale tuturor înregistrărilor al căror titlu conține cuvântul „poezii” și câmpul descriere este gol:

```
UPDATE carti SET
descriere='Carte de poezii'
WHERE titlu like '%poezii%'
and descriere='';
```

Astfel, toate cărțile fără descriere al căror titlu conținea cuvântul *poezii* au acum descrierea *Carte de poezii*.

ALTER TABLE

ALTER TABLE ne permite să schimbăm structura unui tabel existent. Putem

adăuga sau șterge coloane și indecsi, putem redenumi coloane sau chiar tabelul în sine și putem schimba tipul de date conținut de o coloană. Să adăugăm o coloană, **data**, în care să stocăm data adăugării cărții în baza de date. Câmpul **data** al tabelului **carti** ne va fi util atunci când vom afișa vizitorilor noutățile din magazin pe prima pagină. El nu corespunde datei la care a fost editată cartea ci datei în care aceasta a ajuns pe rafturile virtuale ale magazinului nostru. Cele mai noi 10 titluri din librărie le-am afla astfel foarte ușor folosind următoarea interogare:

```
SELECT titlu FROM carti ORDER BY
data DESC LIMIT 0,10;
```

Să adăugăm noua coloană, la sfârșitul tabelului:

```
ALTER TABLE carti ADD dat TEXT;
```

Am scris gresit numele coloanei, să o redenumim din „dat” în „data”:

```
ALTER TABLE carti
CHANGE dat data TEXT;
```

Ne aducem aminte că există tipul DATE și că o dată cum e 12-12-2003 ne-ar putea fi mai utilă dacă s-ar afla într-o coloană de tip DATE decât într-o de tip TEXT așa că vom modifica tipul de date al noii coloane:

```
ALTER TABLE carti
CHANGE data data DATE;
```

Ca să adăugăm o coloană altundeva decât la sfârșit, între descriere și dată spre exemplu, folosim ALTER TABLE în felul următor:

```
ALTER TABLE carti ADD
pret MEDIUMINT UNSIGNED
AFTER descriere;
```

INDECȘI

Cel mai folosit tip de index este id-ul. Id-ul este un număr unic de identificare pentru un element distinct (un rând) al unui tabel. Având structura tabelului **carti** în forma actuală, pentru a alege „Romeo și Julieta” de William Shakespeare, interogarea ar arăta în felul următor:

```
SELECT * FROM carti WHERE
autor='William Shakespeare'
AND titlu='Romeo și Julieta';
```

Ce ne facem însă dacă implementăm interogarea în această formă în aplicație și într-o bună zi primim spre vânzare aceeași carte dar cu un preț diferit? Ar trebui să rescriem interogarea pentru a ne adapta situației și atunci căutarea unei cărți ar fi făcută cu interogarea:

```
SELECT * FROM carti WHERE
autor='William Shakespeare'
AND titlu='William Shakespeare'
AND pret=100000;
```

Dacă am folosi un câmp care să conțină un număr unic de identificare pentru fiecare carte căutarea ar fi mult simplificată:

```
SELECT * FROM carti WHERE
id_carte=15;
```

Un exemplu de id din viață reală îmi vine în minte: ați fost vreodată la un centru de închiriat casete video? Vă amintiți ordinea în care erau puse casetele în rafturi? Un patron de centru de închirieri numerotează fiecare casetă nou primită și o pune la coadă, în ordinea numerotării, nu va căuta să vadă unde ar intra din punct de vedere alfabetic, mai ales că titluri precum „The Firm” ar putea fi puse și sub litera F și sub litera T, după cum consideră fiecare, și atunci confuzia ar fi și mai mare. În plus ca să adauge titluri noi în lista de casete disponibile nu va trebui să rescrie toată lista, ci va adăuga doar titlurile noi la coadă, în ordinea numerotării. Dumneavoastră puteți astfel să aflați rapid ce e nou doar aruncând o privire pe ultima pagină a ofertei. Atunci când cereți o casetă într-un centru de închirieri specificați vânzătoarei numărul sub care se găsește titlul în listă, altfel ei i-ar lua câteva minute bune să caute un film anume din câteva sute. Într-o bază de date id-ul servește aceste funcții la fel ca în viață de zi cu zi.

Să modificăm tabelul carti astfel încât fiecare carte să aibă un id unic:

```
SET INSERT_ID=#
ALTER TABLE carti ADD id_carte INT
UNSIGNED NOT NULL AUTO_INCREMENT
FIRST, ADD INDEX (id_carte);
```

SET INSERT_ID=# este folosit ca pentru fiecare carte deja adăugată să fie pus un id unic. De asemenea, pentru toate cărțile ce le veți adăuga de acum înainte va exista un id unic, incrementat automat.

Să ne folosim de index pentru a executa operații în tabel:

```
INSERT INTO carti(titlu) values('un
text oarecare');
SELECT id_carte, titlu FROM carti;
```

Valoarea id_carte a acestei ultime înregistrări este 9. O folosim pentru a efectua modificări:

```
UPDATE carti SET titlu='un text
modificat' WHERE id_carte=9;
SELECT id_carte, titlu FROM carti
WHERE id_carte=9;
```

Rezultatul va fi:

	id_carte	titlu
+	+	+
	9	un text modificat
+	+	+

Normalizarea

Normalizarea înseamnă structurarea tabelelor bazei de date în aşa fel încât datele din acestea să ocupe cât mai puțin spațiu pe hard disc, fiind astfel deosebit de utilă. Chiar dacă spațiul nu poate părea o problemă, trebuie să luați în considerare posibilitatea de a fi nevoie să vă extindeți într-o zi peste numărul de megabiți oferit de contractul încheiat cu ISP-ul, din lipsă de spațiu. De ce să plătiți mai mult când puteți să normalizați? Normalizarea este o practică bună și vă recomand să o luați în calcul ori de câte ori faceți o bază de date.

Să studiem pentru un moment tabelul carti al bazei noastre de date. În momentul de față el este structurat în felul următor.

Field	Type
autor	text
titlu	text
descriere	text
pret	mediumint(8) unsigned
data	date

Ce s-ar întâmpla dacă am avea toate cărțile scrise de Isaac Asimov? Numărul acestora depășește 400 și coloana **autor** ar conține de tot atâtea ori textul „Isaac Asimov”. Ce putem face este să creăm un nou tabel pentru autori cu două câmpuri: **id_autor** și **nume_autor** și să înlocuim câmpul text **autor** din tabelul cărți cu un câmp numit tot **id_autor** în care pentru toate cărțile unui autor să punem id-ul din tabelul de autori. În acest mod câmpul va conține un număr din câteva cifre în loc de

un text mult mai lung. Dacă scriem într-un document text numele „Isaac Asimov” vedem că acesta ocupă 12 bytes în timp ce un număr din 3 cifre (presupunând că id-ul autorului ar fi compus din 3 cifre) ocupă doar 3. Astfel înlocuind elementele repetitive dintr-un tabel cu un id facem o economie serioasă de spațiu.

Să facem, deci, un nou tabel: **autori**,

```
CREATE TABLE autori
(id_autor SMALLINT UNSIGNED
DEFAULT '0' NOT NULL
AUTO_INCREMENT,
nume_autor TEXT,
PRIMARY KEY(id_autor),
UNIQUE(id_autor),
INDEX(id_autor));
```

și să redenumim câmpul autor din tabelul carti în id_autor.

```
ALTER TABLE carti CHANGE autor
id_autor TEXT;
```

Nu îi vom modifica deocamdată tipul de date pe care îl conține. Ar trebui să revenim ulterior la tabelul autori, să căutăm id-ul corespondent fiecărui autor și să modificăm manual fiecare câmp în parte.

Vom reduce numărul de operații necesare cu ajutorul funcției UPDATE. Adăugăm întră autorul în noul tabel, aflăm id-ul unic repartizat de către MySQL și apoi înlocuim în tabelul carti numele autorului cu acest id.

```
INSERT INTO autori values('',
'Mihai Eminescu');
SELECT * FROM autori WHERE
autor='Mihai Eminescu';
```

id_autor	nume_autor
1	Mihai Eminescu

```
UPDATE carti SET id_autor=1 WHERE
id_autor='Mihai Eminescu';
INSERT INTO autori values('',
'George Cosbuc');
SELECT * FROM autori;
```

id_autor	nume_autor
1	Mihai Eminescu
2	George Cosbuc

```
UPDATE carti SET id_autor=2 WHERE
id_autor='George Cosbuc';
```

Temă: procedați în același fel cu fiecare nume de autor din tabelul cărți.

Acum că avem id-uri numerice în loc de nume în coloana id_autor, să modificăm tabelul **carti** și să transformăm tipul câmpului id_autor din TEXT în SMALLINT:

```
ALTER TABLE carti CHANGE id_autor
id_autor SMALLINT NOT NULL;
```

Cele două tabele arată acum așa:

autori:

id_autor	nume_autor
1	Mihai Eminescu
2	George Cosbuc
3	William Shakespeare
4	Alexandru Mitru

carti:

id_autor	titlu
3	Hamlet
1	Poezii
1	Poezii, volumul 2
1	Poezii, volumul 3
4	Legendele Olimpului
2	Fire de tort - Poezii
1	Geniu pustiu
3	Romeo si Julieta

Putem „uni” datele din cele două tabele astfel încât să ne fie afișate numele și titlul pentru fiecare înregistrare:

```
SELECT nume_autor, titlu FROM
autori, carti WHERE
autori.id_autor=carti.id_autor;
```

nume_autor	titlu
William Shakespeare	Hamlet
Mihai Eminescu	Poezii
Mihai Eminescu	Poezii, volumul 2
Mihai Eminescu	Poezii, volumul 3
Alexandru Mitru	Legendele Olimpului
George Cosbuc	Fire de tort - Poezii
Mihai Eminescu	Geniu pustiu

Această interogare ia coloana nume_autor din tabelul autori și coloana titlu din tabelul cărți și le asociază după id-ul autorului care este unic. Dacă una din coloanele pe care vrem să le afișăm are același nume în cele două tabele, o putem apela ca nume_tabel.nume_colonă astfel încât să nu existe ambiguități:

```
SELECT autori.id_autor,
autori.nume_autor, carti.titlu
FROM autori, carti WHERE
autori.id_autor=carti.id_autor;
```

În aplicația noastră va trebui să avem cărțile ordonate după domenii. Nu vom

id_autor	nume_autor	titlu
3	William Shakespeare	Hamlet
1	Mihai Eminescu	Poezii
1	Mihai Eminescu	Poezii, volumul 2
1	Mihai Eminescu	Poezii, volumul 3
4	Alexandru Mitru	Legendele Olimpului
2	George Cosbuc	Fire de tort - Poezii
1	Mihai Eminescu	Geniu pustiu

amesteca romanele de război cu manualele școlare și cu poezile! Ne va trebui în consecință o nouă coloană în tabel, pentru domeniul cărții: Legendele Olimpului vor apartine domeniului „Povești”, cărțile de poezii domeniului Poezii și aşa mai departe.

Temă: modificați tabelul carti pentru a adăuga o coloană **id_domeniu** și creați un nou tabel **domenii** cu două câmpuri: **id_domeniu** și **nume_domeniu**. Adăugați numele de domenii pentru cărțile din baza de date și id-urile acestora în tabelul cărții, conform cu id-urile din tabelul **domenii**.

Temă: afișați titlurile cărților din baza de date și domeniile de care acestea aparțin (ca în exemplul de mai sus, cu tabelul autorii):

nume_domeniu	titlu
Teatru	Hamlet
Poezii	Poezii
Poezii	Poezii, volumul 2
Poezii	Poezii, volumul 3
Povesti	Legendele Olimpului
Poezii	Fire de tort - Poezii

Deocamdată avem în baza de date tabelele **carti**, **autori** și **domenii**. În continuare va trebui să mai adăugăm unul. Ne reamintim că la fiecare carte vizitatorii noștri au posibilitatea să-și poată împărtași impresiile. Pentru aceasta vom crea un nou tabel, comentarii, ale cărui câmpuri nu le voi mai descrie deoarece numele și tipul lor sunt de ajuns declare:

```
CREATE TABLE comentarii (
    id_comentariu INT UNSIGNED
        DEFAULT '0' NOT NULL
        AUTO_INCREMENT,
    id_carte INT
        UNSIGNED DEFAULT '0',
    nume_utilizator TEXT,
    adresa_email TEXT,
    comentariu TEXT,
    PRIMARY KEY
        (id_comentariu),
    UNIQUE (id_comentariu),
    INDEX (id_comentariu));
```

Câmpul **id_carte** din acest tabel ne va

```
default '0',
nr_buc tinyint NOT NULL default
'0');
```

Câmpul **id_tranzactie** corespunde celui din tabelul tranzactii iar câmpul **id_carte** celui din tabelul carti. Câmpul **nr_buc** ne va servi atunci când un utilizator cumpără mai multe bucati din aceeași carte într-o tranzacție și de asemenea pentru a putea face statisticile pentru cele mai vândute cărți. Introduceți câteva înregistrări în tabel folosind **id_tranzactie** din tabelul tranzactii și mai multe **id_carte** din tabelul carti:

```
INSERT INTO vanzari VALUES(1,2,1);
INSERT INTO vanzari VALUES(1,3,1);
INSERT INTO vanzari VALUES(1,4,30);
```

În tranzacția cu **id_tranzacție** 1 s-au vândut: 1 carte cu **id_carte** 2, 1 carte cu **id_carte** 3 și 30 de cărți cu **id_carte** 4.

Am învățat în acest capitol cum să facem o bază de MySQL date și cum să lucrăm cu ea pentru a obține datele de care avem nevoie. Înainte de a trece la capitolul următor însă trebuie să vă spun că pe CD veți găsi câțiva clienti vizuali pentru serverul MySQL cu care vă veți putea construi structura bazei de date mult mai ușor. V-am „chinuit” cu sintaxa în linia de comandă deoarece mai tarziu, când vom lucra cu baza de date prin intermediul PHP, va fi necesar să știm cum se introduc sau se extrag date folosind comenzi SQL.

În continuare vom învăța bazele programării PHP urmând ca apoi să folosim cunoștințele acumulate în aceste două capitoole pentru a obține informații din baza de date prin intermediul PHP și a le afișa utilizatorilor într-o pagină de web.

Mărimea contează?

Cât de mare poate fi un tabel?

MySQL stochează fizic datele unui tabel într-un fișier pe hard disc și cu cât tabelul este mai mare, cu atât mărimea acestui fișier crește. Versiunea 3.22 a MySQL are o limită de 4GB pentru mărimea unui tabel. În versiunile superioare această limită este extinsă până la 8 milioane TB pentru tipul de tabel MyISAM. Cu toate acestea, sistemele de operare pot avea propriile limitări ale mărimii fișierelor. Mărimea implicită a tabelelor MySQL este de aproximativ 4GB. Puteti verifica mărimea maximă pentru un tabel cu ajutorul comenziilor SHOW TABLE STATUS sau myisamchk -dv table_name.

Pe platforma Windows va trebui să folosiți sistemul de fișiere NTFS dacă dorîți să aveți tabele mai mari de 4GB.

folosi pentru a afla toate comentariile aferente unei cărți, cu ajutorul interogării

```
SELECT * FROM comentarii WHERE
id_carte=1;
```

Pentru urmărirea vânzărilor vom folosi două tabele, tranzactii și vanzari. Să presupunem că Pop Ion cumpără două cărți în 1 martie și revine pentru a cumpăra încă alte trei cărți în 2 martie. În data de 3 martie când vin banii pentru prima comandă noi vom ști că trebuie să trimitem niște cărți lui Pop Ion dar nu știm exact care! Iată de ce, pentru fiecare comandă, indiferent de numărul cărților cerute, va trebui să folosim un număr unic de identificare, număr pe care îl ținem în tabelul tranzactii alături de celelalte informații care ne interesează:

```
CREATE TABLE tranzactii (
    id_tranzactie INT unsigned
        NOT NULL auto_increment,
    data_tranzactie TIMESTAMP(10)
        NOT NULL,
    nume_cumparator TEXT NOT NULL,
    adresa_cumparator TEXT NOT NULL,
    UNIQUE KEY id_tranzactie
        (id_tranzactie));
```

Am setat câmpul **data_tranzactie** ca **TIMESTAMP(10)** pentru că astfel el va putea fi setat automat de MySQL la fiecare nouă înregistrare și va conține data inserării înregistrării în format YYYYMMDD. Executați următoarele două interogări:

```
INSERT INTO tranzactii
    (nume_cumparator,
    adresa_cumparator) VALUES ('Pop
    Ion', 'str. Lunii 12, Brasov');
SELECT * FROM tranzactii;
```

Tabelul vânzări va conține informațiile despre cărțile vândute în fiecare tranzacție:

```
CREATE TABLE vanzari (
    id_tranzactie int unsigned NOT
    NULL default '0',
    id_carte int unsigned NOT NULL
```

PHP

Programare pentru toti

Ca limbaj de programare, PHP este foarte ușor de învățat dacă elementele programării sunt explicate pe înțelesul tuturor. Acesta este unul din scopurile articolelor de față: de a iniția orice novice în tainele programării PHP. Datorită similarității între limbajele de programare de uz comun, cunoscând PHP va fi floare la ureche să treceți apoi la VBScript, Java sau chiar C. Nu i-am uitat pe programatorii experimentați în alte limbaje care vor afla în acest capitol particularitățile PHP.

PHP este limbajul ideal pentru construirea de pagini web dinamice. Este ușor de învățat, open-source, poate fi rulat pe mai multe platforme și se poate conecta la mai multe tipuri de baze de date.

Cel mai important aspect al limbajului este însă posibilitatea de a fi îmbricat cu cod HTML. Putem astfel crea pagini HTML statice și din loc în loc, acolo unde este nevoie, să introducем dinamism cu ajutorul PHP. Să luăm prima pagină a site-ului www.chip.ro de exemplu. Mare parte din structura sa este compusă din cod HTML static (meniurile, tabelele, aranjamentul în pagină). Din loc în loc codul HTML este intercalat cu cod PHP care extrage din baza de date cele mai noi știri. În momentul în care apelați pagina, acest cod este parsat (analizat linie cu linie și executat) pe server și este afișată o pagină HTML fără să știți că pentru crearea ei s-a făcut o conexiune la baza de date, s-au extras informațiile de acolo și au fost ordonate pentru afișare.

Pe locuri...

Limbajul PHP s-a „născut” în 1994 din nevoia lui Rasmus Lerdorf de a afla câte persoane îi vizualizează CV-ul online. El a denumit setul de scripturi create PHP, acronimul pentru Personal Home Page. Pe parcursul următorilor trei ani limbajul a evoluat dar adevăratul succes a început să îl cunoască de când Zeev Suraski și Andi Gutmans au rescris motorul PHP de la cap la coadă, motor care poartă din versiunea 4 a PHP numele Zend, o combinație de litere din prenumele creatorilor săi: Zeev și Andi.

Fiind open-source, PHP beneficiază de suport activ din partea comunității online, acesta fiind și motivul creșterii

explosive a numărului site-urilor bazate pe PHP. Între 2000 și 2002 numărul lor a crescut cu peste 13130% în timp ce numărul site-urilor bazate pe tehnologia ASP doar cu 278%, Java Server Pages cu 1594% și ColdFusion cu 429%.

Pe lângă manipularea conținutului paginilor de web, PHP poate trimite headere HTTP pentru autentificare, seta cookie-uri sau redirecționa utilizatorii. Mai mult, cu ajutorul bibliotecilor externe de funcții poate parse fișiere XML, crea și manipula imagini, animații Shockwave Flash, PDF-uri sau se poate conecta la un server de mail iar acestea sunt doar câteva din funcțiile pe care le poate îndeplini.

Fiți gata...

În continuare vă voi îndruma prin elementele de bază ale PHP. Așezați-vă confortabil, deschideți un editor de text, asigurați-vă că serverul web este pornit și punteți-vă centurile de siguranță. Vă recomand să rulați chiar voi exemplele prezente în continuare deoarece să veți înțelege foarte ușor cum funcționează PHP la server și care este rezultatul în browser.

Puteți folosi orice editor de text. NotePad este bun pentru început dar puteți alege să folosiți unul din editoarele PHP de pe CD. Salvați fișierele cu extensia php, în document root (c:\Program Files\Apache Group\Apache\htdocs) și le veți accesa în browser la adresa <http://localhost/numefisier.php>.

Start!

Programarea de orice fel, nu doar PHP, are două elemente de bază: datele și instrucțiunile. Pentru a lucra cu datele trebuie să înțelegem ce sunt variabilele și

tipurile iar pentru a lucra cu instrucțiuni trebuie să aflăm ce sunt structurile de control și funcțiile.

Variabile

O variabilă este o zonă de memorie căruia îi se dă un nume pentru a putea fi recunoscută ulterior și pentru a ne putea referi mai târziu la ea. Să ne imaginăm ca Ionel și Gigel au următoarea conversație:

Ionel: Poți aduna două numere?

Gigel: Da. Care este primul număr?

Ionel: Primul număr este 3. Al doilea număr este 2.

Gigel: Rezultatul adunării este 5.

Toate cele trei numere din conversație de mai sus le putem considera variabile. Gigel ține minte valoarea primului număr, 2, apoi ține minte valoarea celui de-al doilea număr, 3, după care le adună și obține rezultatul: 5. Discuția ar putea continua în felul următor:

Ionel: Cel de-al doilea număr este acum 6. Care este rezultatul adunării?

Gigel: Este 9.

Ionel a schimbat valoarea celui de-al doilea număr, rezultatul fiind de această dată 9. Putem considera numerele respective ca fiind variabile și atunci codul PHP pentru prima adunare ar fi:

```
$primul_numar = 3;
$al_doilea_numar = 2;
$total = $primul_numar +
        $al_doilea_numar;
```

Valoarea variabilei \$total este suma celor două numere, 5 în acest caz. La o două operație de adunare Gigel ține minte deja valoarea primului număr, schimbă valoarea celui de-al doilea și obține un nou rezultat:

```
$al_doilea_numar = 6;
$total = $primul_numar +
        $al_doilea_numar;
```

Valoarea variabilei \$total se schimbă, deoarece am schimbat unul din elementele ecuației. Înțelegem acum de ce sunt numite variabilele așa: pentru că valoarea lor poate fi schimbată ori direct, cum am schimbat în o două operație valoarea lui \$al_doilea_numar din 3 în 6, ori ca

urmare a dependenței lor de alte variabile, așa cum se schimbă valoarea variabilei \$total în funcție de valorile lui \$primul_numar și \$al_doilea_numar.

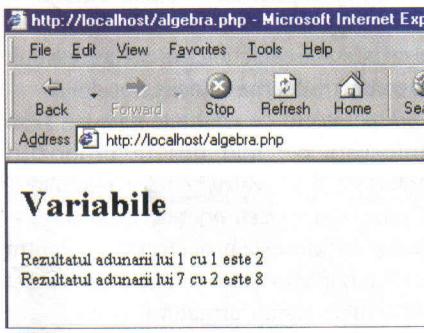
Pentru a înțelege mai bine conceptul de variabilă, putem face o comparație cu algebra. Dacă luăm un număr x și îl dăm valoarea 1, $x+1=1$, $x+2=2$. Dacă $x=7$, $x+1=8$ și $x+2=9$. Analog, putem scrie următorul cod PHP într-un fișier numit algebra.php:

```
<html>
<body>
<h1>Variabile</h1>

<?
// să setăm valoarea lui x ca fiind 1
$x = 1;
$rezultat = $x+1;
print "Rezultatul adunării lui $x cu 1 este $rezultat<br>";

/* și acum să setăm valoarea lui $x ca fiind 7 și îl vom aduna tot cu 1 */
$x = 7;
$rezultat = $x+1;
print "Rezultatul adunării lui $x cu 2 este $rezultat<br>";
?>
</body>
</html>
```

Salvați fișierul în document root (directorul c:\Program files\Apache Group\Apache\htdocs) și accesați <http://localhost/algebra.php>. Iată cum va arăta pagina afișată:



Rezultatul rulării în browser a fișierului algebra.php

Să disecăm codul pentru a învăța câteva lucruri:

– toate instrucțiunile PHP se termină cu punct și virgulă. Omiterea semnului „punct și virgulă” este cea mai frecventă greșală pe care o fac programatorii începători.

– codul PHP începe întotdeauna cu <? și se termină cu ?>. El poate fi imbricat cu

cod HTML după cum puteți vedea în exemplul alăturat. Puteți chiar crea pagini HTML fără pic de cod PHP în ele și să le dați extensia php. Atâtă vreme că parser-ul PHP nu „vede” tagurile <?...?>, el va trimite pagina HTML neschimbată către server.

– putem pune diacritice în cadrul unui string însă pentru ca browserul să le afișeze corect va trebui să specificăm în <head> setul de caractere folosit, la fel ca într-un document HTML.

– valoarea unei variabile poate fi schimbă după necesitate (așa cum am schimbat valoarea lui \$x) sau ea va fi schimbă automat în funcție de celelalte variabile de care depinde (așa cum valoarea lui \$rezultat s-a schimbat în funcție de valoarea lui \$x).

– pentru a afișa rezultatul folosim print, altfel valorile variabilelor ar fi fost schimbate dar nu ar fi fost afișate pe ecran. Pentru mai multe detalii privind folosirea lui print vă recomand să consultați oglinda alăturată.

– linia de text care începe cu // nu este afișată, la fel ca textul demarcat de /* ... */ și nu apar nici măcar dacă dăm View Source în browser pentru fișierul algebra.php accesat. Acestea sunt comentarii care nu sunt procesate de către server ca fiind cod executabil și nici nu sunt trimise mai departe către browser. În aplicațiile mai mari de câteva lini este util să comentăm codul pentru a ne orienta mai bine sau a explica acțiunile întreprinse. Diferența între cele două notății este că // este folosit pentru a comenta o singură linie de text în timp ce /* ... */ poate fi folosit pentru a delimita un comentariu ce se extinde pe mai multe lini. Marcatorii de comentariu se pot folosi și atunci când dorim ca o bucată de cod să nu ruleze. Puteți să testați acest lucru comentând una din liniile care conțin instrucțiunea print și rulând din nou pagina.

Numele variabilelor trebuie să conțină doar litere (a-z, A-Z și caracterele ASCII de la 127 la 255), cifre și liniuțe de subliniere (underscores) și pot începe doar cu litere sau liniuțe de subliniere. Astfel, \$o_variabila, \$_altaVariabila și \$_inca_o_variabila_sunt

Folosirea funcției print

Afișarea datelor

Cu ajutorul lui print putem afișa un string, o variabilă, un string ce conține variabile sau rezultatul unei funcții. Textul ce urmează a fi afișat trebuie inclus între ghilimele simple sau duble, rezultatul fiind ușor diferit. Folosind ghilimele duble orice variabilă din cadrul stringului este parsată. Astfel, în exemplul alăturat, print "Rezultatul inmultirii lui \$x cu 1 este \$rezultat
"; afișează în browser Rezultatul inmultirii lui 1 cu 1 este 2. Dacă am fi folosit ghilimele simple (print "..."), variabilele \$x și \$rezultat nu ar fi fost parsate și output-ul ar fi fost Rezultatul inmultirii lui \$x cu 1 este \$rezultat. Testați acest lucru!

Putem afișa variabilele și dacă folosim ghilimele simple, „rupând” stringul și intercalându-l cu variabile, în forma următoare, folosind operatorul „.” de concatenare a stringurilor (șirurilor):

```
print 'Rezultatul inmultirii lui '.$x.' cu 1 este '.$rezultat.'<br>'
```

Pentru a afișa caracterele speciale folosite de PHP (ghilimelele duble „”, ghilimelele simple „, backslash-ul \, semnul \$) trebuie să le precedeți cu semnul \, astfel:

```
print "Semnul dollar \$, back-slash \\";
```

Ghilimelele trebuie precedate de semnul \ doar dacă sunt de același tip cu cele care încadrează string-ul. În plus, într-un string puteți folosi celelalte ghilimele normale.

print "Ghilimelele duble \" dintr-un string încadrat tot de ghilimele duble trebuie precedate cu semnul \\. Ghilimelele simple 'ca acestea' nu au nevoie să fie precedate deoarece se află într-un string încadrat de ghilimele duble";

Alternativ:

print 'Ghilimelele simple trebuie \'precedate\' într-un string încadrat de ghilimele simple în timp ce "ghilimelele duble" nu';

Dacă dorîți să afișați doar valoarea unei variabile, puteți să nu o încadrați între ghilimele.

```
<?
$o_variabila = 1;
print $o_variabila;
$alta_variabila = "Un text oarecare";
print $alta_variabila;
?>
```

Observați că dacă rulăm acest cod, rezultatul afișat în browser va fi „Un text oarecare”, deși instrucțiunile de afișare se află pe două liniile diferite. Aceasta se întâmplă deoarece, cum ziceam și în introducere, rezultatul este trimis către browser ca HTML. Dacă vrem să ne fie afișate una sub alta va trebui să intercalăm cod HTML între cele două instrucțiuni. Putem face acest lucru în două moduri: ori prin întreruperea codului PHP, așa:

```
<?
$o_variabila = 1;
print $o_variabila;
?>
<br>
<?
$alta_variabila = "Un text oarecare";
print $alta_variabila;
?>
ori prin scrierea lui &lt;br&gt; direct în cod, așa:
<?
$o_variabila = 1;
print $o_variabila;
print "<br>";
$alta_variabila = "Un text oarecare";
print $alta_variabila;
?>
```

nume corecte în timp ce \$o-noua-variabila, \$2var și \$a_saptea nu sunt. Înainte de a trece mai departe, vă dau un sfat: dați variabilelor nume care să definească pe cât posibil destinația variabilei, mai ales dacă aceasta va fi folosită de mai multe ori în cadrul programului. Dacă vi se va cere să modificați aplicația după două luni de la crearea ei o să preferați să înghițeți tastatura mai degrabă decât să vă chinuiți să aflați din 500 de linii de cod ce reprezintă \$a1 sau \$xGvHKx.

Tipuri de variabile

Variabilele pot fi de mai multe tipuri, nu doar numere cum am văzut până acum. Dacă programarea PHP ar fi fost despre algebră și numere, n-aș fi fost aici povestindu-vă cât de minunată este! PHP are opt tipuri de variabile dintre care vi le voi descrie pe cele mai importante 4.

integer

Variabilele de tip integer sunt numere întregi: 3, 783, -56, 0, -1 sunt valori integer.

string

Un string este o succesiune de caractere (șir). Atunci când sunt folosite în codul PHP stringurile trebuie încadrate între ghilimele și toate caracterele speciale din ele precedate cu semnul \ (detalii și explicații găsiți în oglinda **funcția print**). Iată și câteva exemple de stringuri:

```
$variabila = "un text oarecare";
$variabila = "10 texte oarecare";
$variabila = "10";
```

boolean

Tipul boolean definește o valoare de adevăr, TRUE (adevărat) sau FALSE (fals). Spre exemplu, vom scrie mai târziu o funcție pentru secțiunea de administrare a site-ului cu ajutorul căruia vom verifica dacă utilizatorul este logat ca administrator. După ce va face toate verificările necesare, funcția noastră va returna o valoare de adevăr: TRUE dacă este logat sau FALSE dacă nu este și astfel vom ști dacă să îl acordăm sau nu acces în secțiunea de administrare.

array

Puteți să considerați un array ca fiind o colecție de obiecte (matrice). Pentru a înțelege conceptul, să ne amintim de magazinul de casete video de care vă povesteam într-un capitol anterior. Vă spuneam că fiecare film este indexat cu un număr și astfel

ne putem referi la el atunci când cerem filmul vânzătoarei, folosind numărul respectiv.

Să vedem un exemplu:

```
<?
$filme = array("Casablanca",
"Blairwitch Project",
"Matrix", "Rambo");
/* Nu putem afișa direct valorile
unui array, puteți verifica acest
lucru scriind print $filme; și
rulând scriptul în browser. Putem
însă apela valorile lui, folosind
indexul numeric, care, dacă nu este
definit, începe de la 0. Să cerem
filmul "Casablanca": */
print $filme[0];
/* Pentru a cere "Matrix" scriem:*/
print $filme[2];
/* și pentru Rambo */
print $filme[3];
?>
```

Indexarea unui array poate fi definită de către programator, așa cum și patronul unui magazin de casete video își poate numerota casetele după cum îl place.

```
<?
$filme = array(514=>"Casablanca",
32=>"Blairwitch
Project", 19=>"Matrix",
1024=>"Rambo");
/* Acum caseta cu Rambo are numărul
1024 și aşa ne vom și referi la
ea:*/
.print $filme[1024];
/* Pentru a cere "Matrix" scriem:*/
print $filme[19];
/* și pentru Blairwitch Project */
print $filme[32];
?>
```

Arrayurile pot fi indexate și asociativ, adică putem folosi stringuri în loc de integers pentru a ne referi la valorile unui array:

```
<?
$filme = array("dragoste"=> "Casablanca", "groaza"=>"Blairwitch
Project", "SF"=>"Matrix",
"actiune"=>"Rambo");
/* Să afișăm "Blairwitch Project"*/
print $filme["groaza"];
/*Pentru a afișa "Matrix" scriem:*/
print $filme["SF"];
/* și pentru Rambo*/
print $filme["actiune"];
?>
```

Pentru avansați: în PHP nu este nevoie

să definim întâi variabila sau să-i declarăm tipul. \$i = 1 va fi folosit ca integer sau string în funcție de contextul în care este folosit. Dacă doriți să forțați evaluarea unei variabile ca un anumit tip puteți folosi conversia de tip, astfel: \$i = (int) \$i sau \$i = (bool) \$i. Mai multe detalii despre conversia tipurilor găsiți în manualul PHP de pe CD, în secțiunea Type casting.

Operatori

Cel mai des întâlnit operator este cel de atribuire, definit prin semnul =. Am observat din exemplele de până acum că dacă scriem \$x = 1 nu înseamnă că \$x este egal cu 1 ci că i-s-a acordat valoarea 1. Deși la prima vedere egalitatea și acordarea valorii pot părea a fi același lucru, ele nu sunt! Să considerăm următoarele variabile:

```
$x = 1;
$y = 7;
```

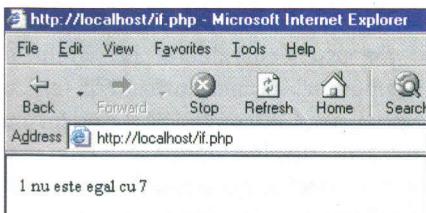
Este \$x egal cu \$y? Nu este. Valoarea lui \$x este 1 în timp ce valoarea lui \$y este 7. Folosind operatorul = de atribuire îi putem acorda lui \$x valoarea lui \$y:

```
$x = 1;
$y = 7;
$x = $y;
```

Abia acum \$x este 7 și este egal cu \$y care are și el valoarea 7. Nu vă lăsați înșelați de asemănarea cu semnul = din matematică nici căci atunci când faceți operații complexe. Dacă scriem \$rezultat = \$x + \$y nu înseamnă că \$rezultat este egal cu suma celor două, ci că i-am atribuit (acordat) valoarea sumei celor două.

Operatorul de egalitate este == și se folosește cel mai des în propoziții condiționale, pentru a testa egalitatea. Opusul său, !=, este operatorul de inegalitate și se folosește în același scop. Pentru a vă lămuri cum să treaba cu egalitatea și atribuirea, testați următorul cod:

```
<?
$x = 1; $y = 7;
if($x == $y)
{
    print "$x este egal cu $y";
}
if($x != $y)
{
    print "$x nu este egal cu $y";
?>
```



Rulați codul și apoi modificați valorile lui \$x și \$y cu \$x = 13 și \$y = 13.

Operatorul de egalitate se folosește pentru a compara egalitatea a două valori. Alți operatori folosiți pentru compararea valorilor variabilelor sunt:

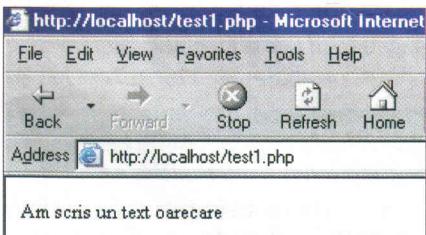
- > mai mare
- >= mai mare sau egal
- < mai mic
- <= mai mic sau egal

```
<?
$x = 5;
$y = 4;
if($x > $y)
{
    print "$x este mai mare ca $y";
}
if($x <= $y)
{
    print "$x este mai mic sau egal cu $y";
}
?>
```

Operatorii pentru stringuri sunt . pentru concatenare și .= pentru atribuirea concatenării. Acești operatori sunt folosiți pentru a uni stringuri, în felul următor:

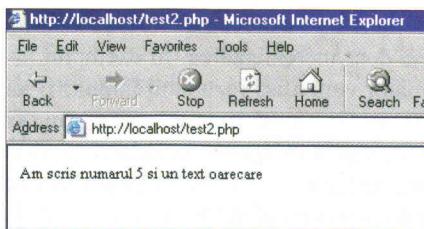
test1.php

```
// concatenarea stringurilor
$test = 'Am scris un '.'text '.'oarecare';
print $test;
// pe ecran va fi afișat „Am scris un text oarecare”
```



test2.php

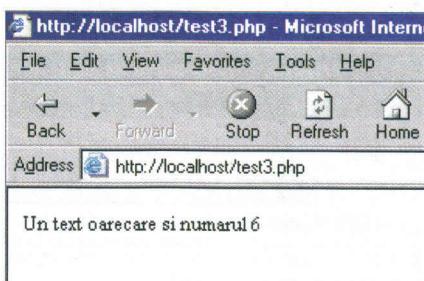
```
// concatenarea stringurilor cu variabile
$nr = 5;
print 'Am scris numărul '.$nr.' și
un text '.'. 'oarecare';
// pe ecran va fi afișat „Am scris numărul 5 și un text oarecare”
```



test3.php

// atribuirea concatenării

```
$nr = 6;
$text = "Un text ";
$text .= "oarecare ";
$text .= "și numărul ".$nr;
print $text;
// pe ecran va fi afișat „Un text oarecare și numărul 6”
```



Operatorii logici vă vor veni la îndemână în execuția scriptului atunci când aveți nevoie să lucrați cu valori de adevăr. Vom face un exercițiu de imaginație pentru a explica funcția și utilitatea operatorilor logici. Să presupunem că la intrarea în secțiunea de administrare avem un formular care cere numele și parola de acces în secțiune. Scriptul PHP ar putea verifica aceste informații pentru a autoriza accesul în secțiune folosind operatorii logici astfel:

– operatorul ! (NOT)

```
if(!parola_e_bună) ... parola nu
este bună, accesul este interzis
if(!parola_nu_e_bună) ... parola e
bună, accesul este permis
```

Operatorul ! returnează TRUE dacă valoarea inițială de adevăr e FALSE și FALSE dacă valoarea inițială este TRUE.

– operatorul || (OR)

```
if(numele_este_valid || parola este
bună) ... verifică dacă numele sau parola sunt valide și dacă oricare din ele este, returnează valoarea de adevăr TRUE. În acest exemplu de pseudocod dacă numele ar fi valid dar parola nu, i-am acordat utilizatorului acces mai departe, ceea ce nu e de dorit. Trebuie să fim siguri că și numele și parola sunt valide.
```

Operatorul || returnează TRUE dacă

oricare din valorile verificate e TRUE. Returnează FALSE doar dacă amândouă sunt FALSE.

– operatorul && (AND)

if(numele_este_valid && parola este
bună) ... dacă atât numele cât și parola sunt valide putem acorda utilizatorului acces în secțiunea de administrare.

Operatorul && returnează TRUE doar dacă ambele valori verificate sunt TRUE. El returnează FALSE dacă oricare din ele este FALSE (sau dacă amândouă sunt FALSE).

Structuri de control

Structurile de control sunt instrucțiunile care aduc flexibilitatea în programare și ușurează munca programatorului.

Am văzut cum putem defini o variabilă \$x și cum putem obține ca rezultat o altă variabilă \$total = \$x + 1. Putem, cu ajutorul structurilor de control să manipulăm variabilele și rezultatele cu minim de cod. Vom vedea cum, cu ajutorul lui while și for putem folosi aceeași bucată de cod pentru mai multe variabile fără să trebuiască să rescriem codul pentru fiecare valoare a variabilei. Vom învăța și structurile if ... else if ... else și switch cu care vom putea executa cod doar dacă sunt îndeplinite anumite condiții.

while

Structura de control while este folosită pentru a rula același cod pentru mai multe valori ale unei variabile oarecare \$x, fără să trebuiască să rescriem codul pentru \$x = 1, \$x = 2 și aşa mai departe pentru fiecare valoare a variabilei. Să folosim pentru testare un cod asemănător celui cu care am pornit la începutul capitolului și să vedem cum structurile de control ne pot ușura munca.

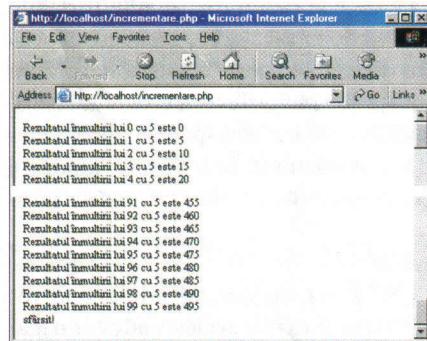
```
<?
$x = 1;
$resultat = $x*5;
print "Rezultatul înmulțirii lui $x cu
5 este $resultat<br>";
$x = 2;
$resultat = $x*5;
print "Rezultatul înmulțirii lui $x cu
5 este $resultat<br>";
?>
```

Să presupunem că ar trebui ca pentru toate numerele de la 0 la 99 ar trebui să

calculăm variabila \$rezultat și să afișăm textul „Rezultatul înmulțirii lui \$x cu 5 este \$rezultat”. Dacă ar fi să scriem codul de înmulțire și afișare pentru fiecare valoare a lui \$x de la 0 la 99 ne-ar apuca dimineața. Pentru aceasta vom folosi while, cel mai simplu tip de loop și vom scrie o singură bucătă de cod care va prelua automat toate valorile lui \$x de la 0 la 99 și pentru fiecare va înmulți cu 5 și va afișa rezultatul automat. Să vedem și cum:

```
<?
$x = 0;
while ($x < 100)
{
    $rezultat = $x*5;
    print " Rezultatul înmulțirii lui $x
        cu 5 este $rezultat<br>";
    $x++;
}
print "sfârșit!";
?>
```

Salvați acest cod într-un fișier incrementare.php și accesați-l din browser pentru a vedea rezultatul.



Înmulțirea automată a numerelor de la 0 la 99 cu 5.

Să vedem, pas cu pas, cum funcționează while. Întâi am definit valoarea inițială a lui \$x. Apoi, pentru fiecare valoare a lui \$x mai mică strict decât 100 (0, 1, 2, 3... până când \$x ajunge la 99) se calculează variabila \$rezultat și se afișează textul, iar în final \$x este incrementat (adunat) cu 1, căpătând astfel o nouă valoare, și programul o ia din nou de la capăt cu \$x = 2, apoi cu \$x = 3 și tot așa până când \$x ajunge la 100. Când \$x ajunge la 100 condiția din while, \$x < 100, nu mai este îndeplinită așa că PHP încetează rularea loop-ului și continuă cu restul de cod. \$++ este o expresie puțin mai specială și este echivalentă cu incrementarea cu 1. \$x++ este astfel prescurtarea pentru \$x = \$x + 1.

Desigur, putem întreprinde orice

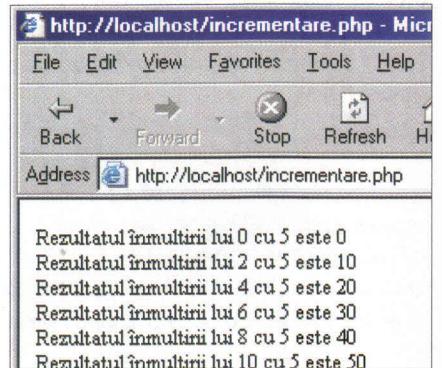
acțiune cu \$x înainte de a relua loop-ul, nu doar incrementarea. În exemplul următor vom afișa rezultatul înmulțirii lui \$x cu 2, apoi îi vom atribui lui \$x valoarea rezultatului înmulțirii și vom repeta loop-ul cu noul \$x până când \$x este mai mic decât 23:

```
<?
$x = 3;
while ($x < 23)
{
    $rezultat = $x *2;
    print "x este $x și rezultatul este
        $rezultat <br>";
    $x = $rezultat;
}
print "sfârșit!";
?>
```

for

for este echivalentul lui while și funcționează în mare măsură la fel, cu câteva mici diferențe. Să rescriem cu ajutorul lui for codul din fișierul incrementare.php pentru a obține aceleși rezultate:

```
<?
for ($x = 0; $x < 100; $x++)
{
    $rezultat = $x * 5;
    print " Rezultatul înmulțirii lui $x
        cu 5 este $rezultat<br>";
    $x++;
}
print "sfârșit!";
?>
```



Structura for în acțiune!

Iată, în românește, ce am făcut: am luat valoarea inițială a lui \$x, 3, și am început loop-ul. La prima rulare se calculează variabila \$rezultat care este \$x * 2 = 3 * 2 = 6 și se afișează pe ecran „x este 3 și rezultatul este 6”. Apoi, folosind operatorul de atribuire „=” în \$x = \$rezultat îi dăm lui \$x valoarea lui \$rezultat care este 6 și loop-ul este reluat de la capăt cu noua valoare \$x = 6.

Pentru \$x = 6 condiția \$x < 32 este îndeplinită, \$rezultat va fi 12 și, la sfârșit atribuim lui \$x această valoare. Pentru \$x = 12 valoarea \$rezultat va fi 24, valoare atribuită din nou variabilei \$x. Cu \$x = 24 condiția pentru rularea loop-ului, \$x < 23, nu se va îndeplini și PHP va trece mai departe, la instrucțiunea print „sfârșit!”. Vă recomand să testați acest cod personal.

Să mai observăm un lucru înainte de a trece mai departe: codul ce urmează a fi executat este încadrat de acolade și acolada de închidere nu este urmată de punct și virgulă. Încadrarea în acolade servește pentru recunoașterea bucătii de cod ce urmează a fi rulată în cazul unei structuri de control. În unele cazuri, vom vedea, acoladele pot fi omise.

Structura for este preferată de programatori deoarece între cele două paranteze ce o precedă sunt conținute toate instrucțiunile și condițiile necesare rulării loop-ului. Astfel, în (\$x = 0; \$x < 100; \$x++), prima parte, \$x = 0 reprezintă situația inițială de la care se pornește rularea loop-ului. A doua parte, \$x < 100, reprezintă condițiile ce trebuie îndeplinite pentru ca loop-ul să fie rulat. Puteți rula următorul cod ca să observați că loop-ul nu va fi rulat deoarece condiția ca \$x să fie mai mic decât 99 nu este îndeplinită.

```
for ($x = 1000; $x < 99; $x++)
{
    print $x;
}
```

În sfârșit, a treia parte o constituie acțiunile ce urmează a fi întreprinse cu variabila în loop. În exemplul nostru variabila este incrementată cu 1 la fiecare rulare.

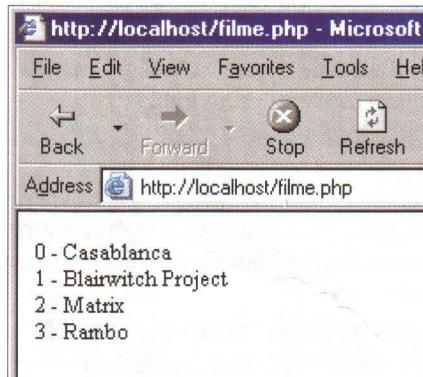
Mai putem adăuga instrucțiuni în această secțiune, separându-le cu virgulă. Iată, spre exemplu, cum am putea scrie toate instrucțiunile pentru manipularea lui \$x în ultima secțiune dintre parantezele lui for:

```
<?
for ($x = 0; $x < 100; print "$x<br>",
    $x++);
print "sfarsit!";
?>
```

Observăm că nici nu mai avem nevoie de acolade. Ele nu ne-ar servi la nimic deoarece toate instrucțiunile privind variabila \$x se află în a treia parte din paranteze: întâi este afișat \$x după care este incrementat și loop-ul o ia de la capăt până când \$x ajunge la 100 și condiția de rulare nu mai este îndeplinită.

Putem folosi for pentru afișarea elementelor unui array cu un index numeric ordonat. Să afișăm lista noastră de filme:

```
<?
$filme = array("Casablanca", "Blairwitch
Project", "Matrix", "Rambo");
$nr_filme = count($filme);
for($i = 0; $i < $nr_filme; $i++)
{
    print "$i - $filme[$i]<br>";
}
?>
```



Rulați scriptul și observați că au fost afișate toate filmele. Cum am făcut acest lucru? Indicându-i lui for de unde să înceapă afișarea elementelor array-ului și unde să se opreasă din loop. Ne-am adus aminte că într-un array, dacă indexul nu este definit, pornește de la 0 și pentru a afișa „Casablanca” trebuie să scriem print \$filme[0] și am făcut același lucru. La a doua rulare a loop-ului \$i a avut valoarea 1 și a afișat filmul \$filme[1] care este „Blairwitch project” și aşa mai departe pentru fiecare element din array.

Funcția count() este folosită pentru a număra elementele dintr-un array. count(\$filme) este 4 în cazul de față.

Teme:

1. Adăugați mai multe filme în array și rulați din nou scriptul.
2. Schimbați \$i = 0 cu \$i = 1 și rulați scriptul.

3. Schimbați \$nr_filme=count(\$filme) cu \$nr_filme = count(\$filme)-1 și rulați din nou scriptul.

if

Dacă dorim să rulăm o bucată de cod doar când este îndeplinită o condiție anume, folosim if. Iată cum, în următorul cod afișăm un text doar dacă numărul de caractere dintr-un string este mai mare decât 5.

```
<?
$text = "Ana are mere";
$nr_caractere = strlen($text);
if($nr_caractere > 5)
{
    print "Textul \"$text\" are mai
          mult de 5 caractere";
}
?>
```

Încercați același cod cu un string mai mic, \$text = "Ana" sau \$text = "mere" pentru a vedea că dacă stringul are mai puțin de 5 caractere textul nu va fi afișat. Observăm folosirea semnului \ de precedare a caracterelor speciale pentru a afișa ghilimelele în rezultatul de pe ecran, astfel: Textul „Ana are mere” are mai mult de 5 caractere.

Am folosit în acest exemplu funcția strlen care calculează numărul de caractere într-un string. Pentru \$text = "Ana are mere", valoarea variabilei \$nr_caractere = strlen(\$text) va fi 12 (spațiile contează). Pentru \$text="mere", \$nr_caractere va fi 4.

Vă spuneam mai sus că în unele cazuri putem să omitem acoladele. În cazul lui if, dacă partea dintre acolade este o singură linie putem „uita” acoladele și scrie codul astfel:

```
if($nr_caractere > 5) print "Textul
\"$text\" are mai mult de 5 caractere";
```

if... else

Ce facem atunci când dorim să afișăm un text dacă o condiție este îndeplinită și alt text dacă aceeași condiție nu este îndeplinită? Folosim if ... else:

```
<?
$text = "mere";
$nr_caractere = strlen($text);
if($nr_caractere > 5)
{
    print "Textul are mai mult de 5
          caractere";
}
```

```
else
{
    print "Textul are mai puțin de 5
          caractere";
}
?>
```

Pentru stringul „mere” rezultatul afișat pe ecran va fi „Textul are mai puțin de 5 caractere” și pentru stringul „Ana are mere” rezultatul va fi „Textul are mai mult de 5 caractere”. Testați acest lucru rulând codul pentru ambele cazuri.

În construcția if ... else putem omite acoladele dacă instrucțiunile ce urmează a fi executate sunt pe o singură linie:

```
if($nr_caractere > 5) print "Textul
are mai mult de 5 caractere";
else print "Textul are mai puțin de 5
caractere";
```

if ... else if ... else

Dar dacă dorim să folosim mai multe condiții și în funcție de fiecare dintre acestea să afișăm altceva? Folosim if ... else if... else și repetăm else if de câte ori avem nevoie.

```
<?
$text = "prune";
$nr_caractere = strlen($text);
if($nr_caractere < 5)
{
    print "Textul are mai puțin de 5
          caractere";
}
else if ($nr_caractere == 5)
{
    print "Textul are exact 5
          caractere";
}
else if($nr_caractere > 12)
{
    print "Textul are mai mult
          de 12 caractere";
}
else
{
    print "Textul are mai puțin de
          5 caractere și mai mult de 12";
}
?>
```

În acest exemplu vedem că else care este folosit chiar la sfârșit, servește pentru toate celelalte cazuri care nu îndeplinesc condițiile specificate în if și else if.

Testați exemplul de mai sus folosind diverse valori pentru \$text.

switch

switch este alternativa pentru blocurile condiționale if ... else if ... else. Este preferabil în cazul în care codul dintre accolade este mai mare și ne-am putea pierde în propoziții condiționale și paranteze. Să vedem cu un exemplu simplu cum se folosește while:

```
<?
$text = "Ana are mere";
$nr_caractere = strlen($text);
switch($nr_caractere)
{
    case 4:
        print "Textul are 4 caractere";
        break;
    case 5:
        print "Textul are 5 caractere";
        break;
    case 12:
        print "Textul are 12 caractere";
        break;
    default:
        print "Textul nu are nici 4 nici
              5 nici 12 caractere";
        break;
}
?>
```

Cazul special „default” de la sfârșit este folosit pentru situația în care dorim să executăm cod și dacă nici una din condițiile anterioare nu este îndeplinită (fiind astfel similar lui else din structura if ... else if ... else). El poate fi omis și astfel să executăm cod doar dacă una din condiții este îndeplinită. Instrucțiunea break; trebuie folosită la încheierea fiecărui caz deoarece altfel codul din următorul caz va fi executat. Iată un exemplu:

```
<?
$litera = "a";
switch($litera)
{
    case "a":
        print "Ana ";
    case "b":
        print "Barbu ";
    case "c":
        print "Claudia ";
}
print "este un nume";
?>
```

Rezultatul rulării acestui cod va fi „Ana Barbu Claudia este un nume” și nu doar „Ana este un nume” așa cum ne-am fi așteptat, deoarece în lipsa instrucțiunii break; la încheierea cazurilor rularea codu-

lui nu s-a oprit. Mai observăm din acest exemplu că în cadrul structurilor de control, de orice fel nu doar switch, putem folosi stringuri dar va trebui să le încadram cu ghilimele (cum nu am fost nevoiți să facem în cazul numerelor). Iată două exemple de folosire a stringurilor într-o structură de control, folosind numere, respectiv stringuri:

```
<?
if ($i == 1)
{
    print "$i este impar";
}
if ($i == 2)
{
    print "$i este par";
}
?>
```

Al doilea exemplu:

```
<?
if($nume == "Ana")
{
    print "Acesta e nume de fată";
}
if ($nume == "Barbu")
{
    print "Acesta este nume de băiat";
}
?>
```

Am învățat până acum ce sunt variabilele și ce fel de structuri de control avem la dispoziție pentru a lucra cu ele astfel încât să obținem rezultatele dorite cu minim de efort. În continuare vom afla ce sunt funcțiile și cum acestea ne fac munca chiar mai ușoară.

Funcții

Vă mai amintiți unde am mai întâlnit termenul de funcție? Dacă vă spun că la algebră o să fugiți mâncând pământul? Ei bine, legați-vă de scaun pentru că acolo am întâlnit termenul și în continuare vă voi explica funcționarea funcțiilor (hmm, asta o fi repetiție?) în PHP. Deși programarea PHP nu este matematică, modul în care tratează funcțiile este similar. Spre exemplu, la algebră, funcția $f(x) = x + 1$ executa o operație cu x și dădea un rezultat în funcție de valoarea lui x . Același lucru îl face și o funcție PHP.

```
<?
// funcțiile trebuie definite înainte
// de a le apela.
```

```
function recalculare($x)
{
    $total = $x + 1;
    print $total;
}
$x = 1;
recalculare ($x);
// va afișa pe ecran 2
?>

<br>Putem de asemenea să folosim direct valoarea variabilei ca parametru (argument) al funcției: <br>
?>
recalculare(16);
// va afișa pe ecran 17
?>
```

Funcțiile pot accepta mai mulți parametri:

```
<?
function inmultire($x, $y)
{
    $rezultat = $x * $y;
}
$x = 5;
$y = 6;
print inmultire($y, $y);
?>
```

Putem de asemenea să dăm parametrii direct, separați prin virgulă
print inmultire(76, 59);
sau aşa:

```
print inmultire($x = 76, $y = 59);
```

Putem da unei variabile valoarea rezultatului unei funcții dacă acesta este o variabilă:

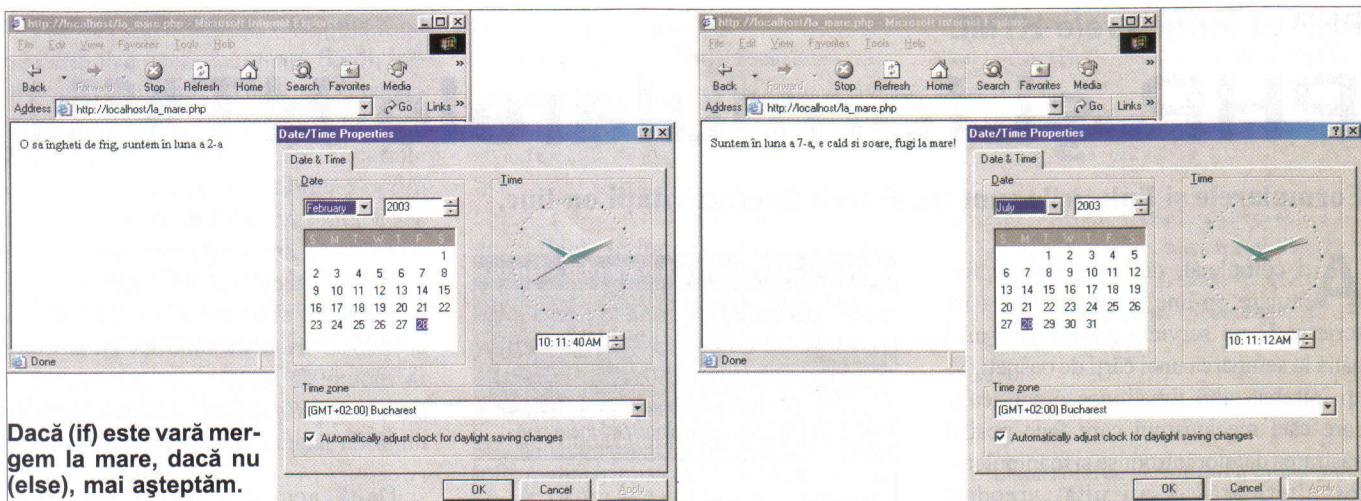
```
$variabila = inmultire($x = 16, $y = 30);
print $variabila;
```

În pasarea unei variabile către o funcție nu contează numele variabilei ci doar valoarea acesteia, valoare ce urmează a fi prelucrată.

```
<?
function adunare($x)
{
    $rezultat = $x + 1;
}
$a = 5;
print adunare($a);
$b = 10;
print adunare($b);
?>
```

În acest exemplu vedem că doar valoarea variabilei este cea care contează, funcția preluând această valoare.

Funcțiile pot fi definite și rulate fără a li se specifica vreun argument:



Dacă (if) este vară mergeam la mare, dacă nu (else), mai aşteptăm.

```
<?
function vreau_la_mare()
{
    $luna = date("n");

    /* date("n") returnează reprezentarea
       numerică a lunii în care suntem, de la 1
       pentru ianuarie până la 12 pentru decembrie.
       Astfel, dacă luna curentă e aprilie,
       $luna va fi 4 și dacă e noiembrie $luna va
       fi 11*/

    if($luna < 6 || $luna > 9) print "O să
        îngehi de frig, suntem în luna a
        $luna-a";
    else print "Suntem în luna a $luna-a,
        e cald și soare, fugi la mare!";
}
vreau_la_mare();
?>
```

Dacă e mai devreme de luna iunie sau (operatorul ||) mai târziu de septembrie, nu e cazul să ne facem bagajele. Altfel, drumul e al nostru!

Puteți modifica (temporar) setările de dată din Windows și să schimbați luna pentru a vedea ce se întâmplă. Nu uitați să reveniți la setările anterioare!

Am văzut din aceste exemple că rezultatul unei funcții poate fi o valoare (funcția înmulțire care oferă rezultatul înmulțirii lui \$x cu \$y) sau o acțiune (funcția vreau_la_mare() care afișează un text). Rezultatul unei funcții mai poate fi și o valoare de adevăr, TRUE sau FALSE, ca în exemplul următor:

```
<?
function e_dimineata()
{
    $ora = date("G");
    /*date("G") returnează ora curentă în
       format 0-24 */
    if($ora >= 5 && $ora <=9)
    {
        return TRUE;
    }
    else
    {
        return FALSE;
    }
}

if(e_dimineata()) print "Poți să faci
    cafeaua!";
else print "E prea târziu pentru cafea.";
```

Sau, același lucru scris folosind

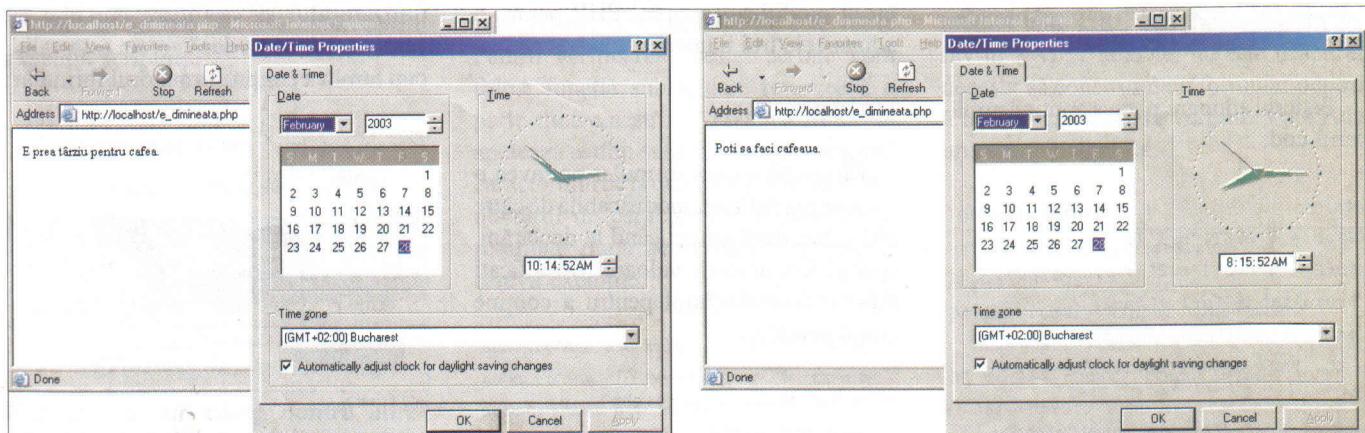
operatorul ! NOT:

```
if(!e_dimineata()) print "E prea
    târziu pentru cafea.";
else print "Poți să faci cafeaua.";
?>
```

Această funcție ilustrează foarte bine folosirea operatorilor logici. Dacă ora este mai mare sau egală cu 5 și mai mică sau egală cu 9, ne putem face cafeaua. Dacă e 4am nu vom bea cafea deși \$ora <=9 și la fel dacă e ora 22 deși \$ora >=5. Ambele condiții trebuie să fie adevărate pentru ca e_dimineata() să fie TRUE.

Funcții predefinite

Toate funcțiile exemplificate până acum au fost definite de noi. Există însă și funcții predefinite, integrate în PHP, pentru o mulțime de acțiuni. Un exemplu este funcția date() pe care tocmai am folosit-o și care returnează ora, luna, anul precum și alte elemente ale datei. Alt exemplu ar fi funcția count() pe care am folosit-o să numărăm câte elemente sunt într-un array. Există funcții pentru aproape orice în PHP și vă sfătuiesc să căutați în manualul PHP de pe CD ori de câte ori aveți nevoie de ceva.



Scriptul care are grija de tensiunea noastră: nici un pic de cafea după ora 10.

PHP și formularele HTML

PHP și formularele HTML

Formularele și link-urile sunt mediatorii interactivității on-line.

Ați completat, desigur, zeci de formulare on-line, începând de la înscrierea la un serviciu gratuit de e-mail până la semnarea unei cărți de oaspeți. V-ați întrebat cum funcționează acestea și care este mecanismul care face posibil păstrarea datelor introduce și manipularea lor ulterioră? V-ați uitat vreodata nedumeriți la link-urile interminabile și aparent fără sens din unele pagini de web?

În acest capitol vom afla cum ajunge informația dintr-un formular la un script PHP și cum, un link poate fi mai mult decât ceea ce pare la prima vedere.

Acțiune!

Un formular este compus în mod normal din declarația <form...> urmată de elemente de tip <input> și un buton „submit”. În momentul în care apăsăm butonul submit, datele adăugate de utilizator sunt transmise serverului unde sunt prelucrate.

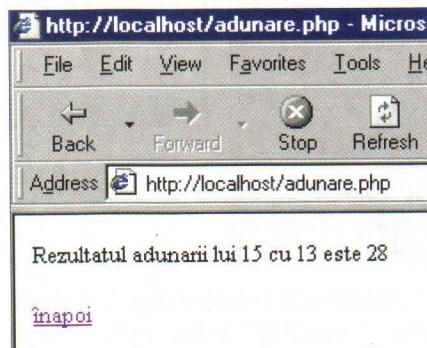
Să vedem practic cum se realizează acest lucru, recreând într-un formular dialogul cu care am început să învățăm PHP, unde Gigel trebuie să adune două numere. Pentru aceasta vom crea două fișiere, formular.html și adunare.php.

În fișierul HTML vom avea următorul formular:

```
<form action="adunare.php" method="POST">
Primul număr: <input type="text"
    name="nr1"><br>
Al doilea număr: <input type="text"
    name="nr2"><br>
<input type="submit"
    name="buton_submit" value="Adună">
</form>
```

Fișierul adunare.php va conține următorul cod:

```
<?
$nr1 = $_POST['nr1'];
$nr2 = $_POST['nr2'];
$rezultat = $nr1 + $nr2;
print "Rezultatul adunării lui $nr1
    cu $nr2 este $rezultat";
?>
<p><a href="formular.html">înapoi
    </a></p>
```



Numerele trimise cu ajutorul formularului au fost adunate și rezultatul afișat.

Rulați <http://localhost/formular.html>, scrieți câte un număr în fiecare căsuță de text și apăsați butonul „Adună”. Repetați experimentul cu alte numere.

Din acest exemplu observăm mai multe lucruri: în declararea tag-ului <form> am specificat fișierul care urmează să prelucreze datele (action="adunare.php"), precum și metoda prin care se transmit datele către server, în cazul de față POST. Elementele formularului au câte un nume după care vor fi recunoscute de către interpretor (ex: \$_POST['nr1']). La apăsarea butonului „Adună”, informația va fi trimisă serverului. \$_POST este un array care conține toate datele trimise prin formular cu metoda POST iar fiecare din elementele acestui array poate fi accesat ca \$_POST['nume_variabilă'].

Temă: adăugați încă un element, <input type="text" name="nr3"> în formular și modificați scriptul PHP pentru a aduna cele trei numere.

Valoare

Elementele unui formular pot avea o valoare predefinită, modificabilă de către utilizator, dacă atunci când le declarăm, specificăm această valoare. Modificați fișierul formular.html pentru a conține codul următor:

```
<form action="adunare.php"
    method="POST">
Primul număr: <input type="text"
```

```
name="nr1" value="15"><br>
Al doilea număr: <input type="text"
    name="nr2" value="13"><br>
<input type="submit"
    name="buton_submit" value="Adună">
</form>
```

Dacă accesăm <http://localhost/formular.html> observăm că în câmpurile destinate numerelor ce urmează a fi adunate există deja valorile 15 și respectiv 13. Putem apăsa butonul „Adună” pentru a obține rezultatul 28 sau putem modifica oricare din valorile prezentate în formular (direct în browser, nu editând codul HTML), pentru a obține alt rezultat.

Verificare

Un utilizator sugubaț ar putea introduce un text în loc de un număr sau ar putea lăsa chiar cele două câmpuri goale. În exemplul nostru aceasta n-ar fi o problemă însă dacă am face o aplicație serioasă on-line pentru primăria orașului, cu ajutorul căreia cetățenii să-și plătească impozitele, omiterea verificării datelor trimise de utilizator ar duce la haos. Dacă cetățenii ar putea scrie un text oarecare în loc de codul numeric personal sau nu l-ar completa, fișa de impozit nu ar putea fi procesată.

De aceea, pentru orice fel de date trimise de către utilizatori vom face verificările de rigoare. Pentru a exemplifică, vom face două fișiere noi, cnp.html cu care vom cere codul numeric personal al utilizatorului și verificare_cnp.php cu care vom verifica dacă acesta este valid. cnp.html va conține următorul formular:

```
<form action="verificare_cnp.php"
    method="POST">
Name:
<input type="text" name="nume"><br>
Cod numeric personal:
<input type="text" name="cnp"><br>
<input type="submit"
    value="Verificare CNP">
</form>
```

Algoritmul din verificare_cnp.php va

fi orientat după cerințe: câmpul „Cod numeric personal” nu poate fi gol, trebuie să fie numeric, să conțină 13 caractere și prima cifră a sa trebuie să fie 1 sau 2. Vom verifica pe rând fiecare din cerințe iar în cazul unei incompatibilități oprim execuția scriptului folosind exit;. Dacă nu am opri execuția după semnalarea erorii, scriptul ar rula în continuare până la capăt și un cod numeric personal invalid ar fi introdus în baza de date a primăriei deși pe pagina de output ar apărea mesajul de eroare. Am putea evita rularea scriptului până la capăt dacă am folosi if ... else:

```
if cerința 1 nu este îndeplinită
{
    afișează mesaj de eroare
}
else
{
    if cerința 2 nu este îndeplinită
    {
        afișează mesaj de eroare
    }
    else
    {
        if cerința 3 nu este îndeplinită...
    }
}
```

Cu if ... else ne-am pierde în acolade așa că vom folosi exit în verificările noastre pentru a opri rularea ulterioară a scriptului dacă una din cerințe nu este îndeplinită.

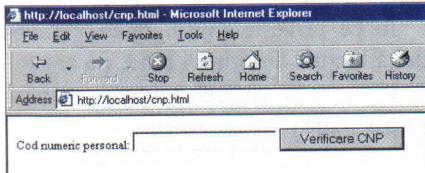
În continuare vom scrie scriptul de verificare a codului numeric personal introdus în formularul cnp.html. Scrieți și voi acest cod omițând comentariile ajutătoare.

<?

/*dacă utilizatorul nu a introdus nimic în câmpul "Cod numeric personal", îi afișăm un mesaj de eroare și oprim execuția scriptului*/

```
if($_POST['cnp'] == "")
{
```

```
    print "Câmpul \"Cod numeric
```



Formular cu un câmp pentru introducerea codului numeric personal pentru verificare.

```
personal\" este gol! Apăsați butonul <b>Back</b> în browser pentru a reveni la pagina anterioară și a-l scrie corect.";
exit;
}

/*
 * Dacă scriptul a ajuns până aici înseamnă că cerința anterioară a fost îndeplinită. Trecem atunci la următoarea verificare: dacă CNP nu este numeric, afișăm alt mesaj de eroare. is_numeric este o funcție predefinită în PHP care verifică după cum îi zice și numele, dacă informația trimisă de utilizator este un număr sau nu*/
if(!is_numeric($_POST['cnp']))
{
    print "Câmpul \"Cod numeric
          personal\" trebuie să fie numeric!
          Apăsați butonul <b>Back</b> în browser pentru a reveni la pagina anterioară și a-l scrie corect.";
    exit;
}

/*
 * dacă CNP nu are exact 13 caractere, afișăm un mesaj de eroare. Funcția strlen returnează numărul de caractere (inclusiv spații) dintr-un string. strlen("mere")=4, strlen("120")=3 și strlen("4 mere")=6.*/
$nr_caractere = strlen($_POST['cnp']);
if($nr_caractere != 13)
{
    print "Câmpul \"Cod numeric
          personal\" trebuie să aibă exact 13
          caractere! Apăsați butonul <b>Back</b> în browser pentru a reveni la
          pagina anterioară și a-l scrie
          corect.";
    exit;
}

/*
 * dacă prima cifră a CNP nu este 1 sau
 * 2, codul numeric personal nu este valid.
 * Pentru verificarea acestei cerințe vom
 * folosi funcția substr care returnează o
 * parte dintr-un string. Funcția substr se
 * apelează astfel: substr(„string”, numărul
 * caracterului de la care se începe “tăierea”,
 * numărul de caractere ce urmează a fi
 * returnate după acest caracter). Iată și
 * câteva exemple:

```

```
substr("Ana are mere", 0, 6) = "Ana ar"
substr("Ana are mere", 1, 6) = "na are"
substr("Ana are mere", 0, 1) = "A"
substr("Ana are mere", 1, 2) = "na"
substr("Ana are mere", 6, 5) = "e mere"
```

```
/*
$prima_cifra = substr($_POST['cnp'],
                      0, 1);
if($prima_cifra != 1 &&
   $prima_cifra != 2)
{
    print "Prima cifra a CNP trebuie
          să fie 1 sau 2! Apăsați butonul
          <b>Back</b> în browser pentru a
          reveni la pagina anterioară și a-
          l scrie corect.";
    exit;
}

/*
*Dacă rularea scriptului a ajuns până
*în acest punct înseamnă că toate cerințele
*au fost îndeplinite și codul numeric personal
*este corect. În aplicația noastră ar
*urma să îl introducem în baza de date dar
*cum nu acesta era scopul exemplului,
*vom afișa doar un mesaj de confirmare:*/
print "Acesta este un CNP valid!";
?>
```

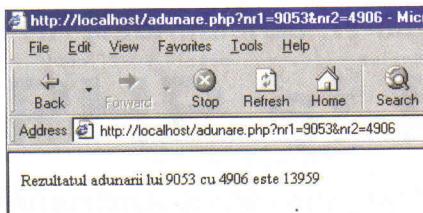
GET

Formularele pot fi transmise serverului și cu metoda GET (<form action="fișier.php" method="GET">) însă este preferabil să evitați să folosiți GET în formulare din trei motive: datele din cadrul formularului sunt transmise serverului prin atașarea la URL, nu se pot transmite decât caractere ASCII și cantitatea de informație transmisibilă este limitată ca mărime.

Dacă folosim metoda GET trebuie ca și în cadrul scriptului PHP să accesăm variabila ca atare (ex: \$_GET['cnp'] în loc de \$_POST['cnp']). Datele formularului sunt transmise prin URL sub forma <http://www.adresa.site/fișier.php?variabilă1=valoare1 & variabilă2=valoare2>. Ca regulă generală, orice variabile trimise scriptului PHP prin intermediul unui URL sunt accesibile în array-ul \$_GET și astfel nu este neapărat necesar să construim un formular pentru a trimite variabile serverului. Vom rula un exemplu simplu:

formular.html

```
<form action="adunare.php"
      method="GET">
    Primul număr: <input type="text"
                           name="nr1"><br>
    Al doilea număr: <input type="text"
                           name="nr2"><br>
    <input type="submit">
```



Variabilele transmise prin metoda GET sunt vizibile în bara de adrese a browserului.

```
</form>

adunare.php
<?
$nr1 = $_GET['nr1'];
$nr2 = $_GET['nr2'];
$rezultat = $nr1 + $nr2;
print "Rezultatul adunării lui $nr1
      cu $nr2 este $rezultat";
?>
```

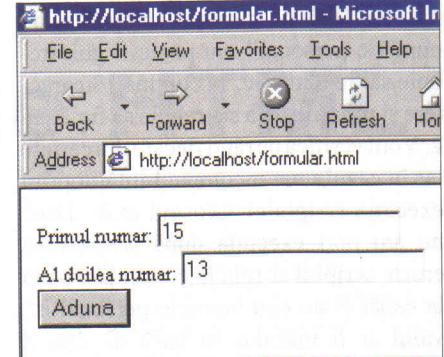
Rulând acest exemplu putem vedea cum numele și valorile elementelor formularului se transmit fișierului PHP în URL. Putem accesa direct în browser <http://localhost/adunare.php?nr1=9053&nr2=4906> sau <http://localhost/>

adunare.php?nr1=423&nr2=546, transmitând datele prin URL, fără să completează formularul.

În cadrul site-ului nostru vom folosi formulare pentru a lăsa posibilitatea utilizatorilor să adauge comentarii și pentru a trimite adresa unde doresc să primească cărțile cumpărate. Datele din formularele completate de către vizitator vor fi trimise serverului prin metoda POST. Tot în cadrul site-ului vom folosi de multe ori GET pentru a obține variabile dintr-un URL. Spre exemplu, pentru a vedea o carte anume adresa va fi http://localhost/carte.php?carte_id=345. Scriptul PHP va selecta din baza de date doar informațiile despre cartea cu id 345 și le va afișa. Vom vedea cum în următoarele două capituloare.

Globals

Există o setare în php.ini care v-ar permite accesul la variabilele trimise de utilizator fără să le accesăm din array. Astfel, într-un form trimis prin metoda POST, valoarea din `<input type="text" name="oarecare">` ar putea fi accesibilă ca `$oarecare` în loc de `$_POST['oarecare']`. Același lucru este vala-



Formularul cu ajutorul căruia vom trimite două numere pentru a fi calculate la server.

bil și pentru metoda GET sau pentru elementele din array-ul `$_FILES`. Variabila `$_GET['oarecare']` ar putea fi accesibilă ca `$oarecare` iar `$_FILES['fisier']['size']` ca `$fisier_size`. Deși poate părea mai ușor de programat așa, este recomandabil să lăsați setarea `Globals=Off` în `php.ini` (aşa cum este setată implicit) și întotdeauna să accesați variabilele din array-ul din care provin. Astfel evitați o serie de probleme de securitate, probleme pe care le vom explica mai pe larg în capitolul dedicat securității aplicațiilor web).

Lucrul cu baze de date

PHP și MySQL

Dezvoltarea explozivă a limbajului PHP din ultimii ani este în cea mai mare parte datorată ușurinței cu care acesta lucrează cu bazele de date.

PHP oferă programatorului o mulțime de funcții predefinite pentru lucrul cu baza de date. Aceasta înseamnă că din cadrul PHP putem executa toate operațiunile pe care le-am făcut în capitolul dedicat învățării MySQL. Să ne reamintim pașii pe care i-am facut pentru a interacționa cu baza de date din linia de comandă:

1. ne-am conectat cu numele și parola (`mysql -u root -p`)
2. am ales baza de date cu care să interacționăm (USE librarie)
3. am executat o comandă SQL (SELECT * FROM carti;)

Din PHP putem face același lucru, cu ajutorul funcțiilor predefinite. Să facem primul nostru script cu care să interacționăm cu baza de date (nu uități că serverul MySQL trebuie să fie pornit!):

test.php

```
<?
mysql_connect("localhost", "root", "");
mysql_select_db("librarie");
mysql_query("SELECT * FROM carti");
?>
```

Observăm similitudinea între cele două

abordări. Cu funcția `mysql_connect` ne conectăm la baza de date specificând hostul la care aceasta se află (localhost), numele de utilizator (root) și parola (pentru că aceasta este goală folosim un string gol: ""). Funcția `mysql_select_db ("librarie")` este echivalentă cu `USE librarie`. Aceste două funcții trebuie apelate întotdeauna atunci când dorim să lucrăm cu baza de date, înainte de a efectua vreo interogare. Putem efectua oricât de multe interogări dorim fără să fie nevoie să ne reconectăm pentru fiecare din ele la baza de date, doar prin apelarea `mysql_connect` și `mysql_select_db` o singură dată la început.

În ce privește `mysql_query`, notăm două lucruri. În primul rând, nu am folosit punct și virgulă pentru a încheia instrucțiunea SQL așa cum era necesar în linia de comandă. În al doilea rând, dacă rulați scriptul în browser nu veți vedea nimic pe ecran așa cum probabil v-ați așteptat. Aceasta pentru că `mysql_query`

execută interogarea dar nu afișează rezultatul ci returnează o valoare: TRUE dacă interogarea a fost efectuată cu succes sau FALSE dacă aceasta a eşuat. Pentru instrucțiunile de tip SELECT, SHOW, EXPLAIN sau DESCRIBE, mysql_query returnează și un identificator de resurse pe care îl putem atribui unei variabile.

Resursele sunt variabile speciale care conțin referințe către resurse externe (precum rezultatul unei interogări a bazei de date) și pot fi manipulate cu ajutorul funcțiilor.

Este evident că dacă rezultatul interogării ar fi fost afișat direct în browser, nu am putea modifica aspectul sau funcția paginilor de web dinamice și amazon.com ar fi o simplă însuruire de date.

Să ne convingem că rezultatul interogării este o resursă. Pentru aceasta modificăm ultima linie a scriptului și scriem:

test.php

```
<?
mysql_connect("localhost", "root", "");
mysql_select_db("librarie");
$resursa = mysql_query("SELECT * FROM
    carti");
print $resursa;
?>
```

Putem de asemenea scrie interogarea SQL într-o variabilă pe care să o folosim ca parametru al funcției mysql_query. Această metodă este preferată de mulți deoarece textul interogării este mai ușor de reperat în cadrul scriptului:

test.php

```
<?
mysql_connect("localhost", "root", "");
mysql_select_db("librarie");
$sql = "SELECT * FROM carti";
$resursa = mysql_query($sql);
print $resursa;
?>
```

PHP oferă funcții pentru accesarea resurselor care rezultă din mysql_query. Cu ajutorul mysql_num_rows putem afla câte rânduri a returnat interogarea:

test.php

```
<?
mysql_connect("localhost", "root", "");
mysql_select_db("librarie");
$resursa = mysql_query("SELECT * FROM
    carti");
$nr = mysql_num_rows($resursa);
print "Sunt $nr cărți în baza de date";
?>
```

Pentru a afișa valori cadrul din resursei returnate putem folosi mysql_result. În exemplul următor afișăm conținutul câmpului titlu de pe primul rând al tabelului (numerotarea începe de la 0):

test.php

```
<?
mysql_connect("localhost", "root", "");
mysql_select_db("librarie");
$resursa = mysql_query("SELECT * FROM
    carti");
$resultat = mysql_result($resursa, 0,
    "titlu");
print $resultat;
?>
```

Parametrii funcției mysql_result sunt: resursa (\$resursa), rândul (0, primul rând al tabelului) și numele câmpului ("titlu");

test.php

```
<?
mysql_connect("localhost", "root", "");
mysql_select_db("librarie");
/* afișează conținutul câmpului descriere de pe cel de-al doilea rând al tabelului */
$resursa = mysql_query("SELECT * FROM
    carti");
$resultat = mysql_result($resursa, 1,
    "descriere");
print $resultat;
/* sau conținutul câmpului id_carte de pe cel de-al treilea rând al tabelului*/
print mysql_result($resursa, 2,
    "id_carte");
?>
```

Mysql_result este greoi de folosit deoarece suntem nevoiți să accesăm fiecare coloană a fiecărui rând în parte. În ajutorul nostru vine mysql_fetch_array cu care putem accesa valorile din tabelul returnat în interogare dintr-un array. Putem accesa resursa ca un array numeric:

test.php

```
<?
mysql_connect("localhost", "root", "");
mysql_select_db("librarie");
$resursa = mysql_query("SELECT *
    FROM carti");
$arrRezultat =
    mysql_fetch_array($resursa, MYSQL_NUM);
print_r ($arrRezultat);
?>
```

sau asociativ:

```
$resursa = mysql_query("SELECT *
```

```
FROM carti");
$arrRezultat =
    mysql_fetch_array($resursa, MYSQL_ASSOC);
print_r ($arrRezultat);
```

Dacă nu specificăm tipul array-ului sau folosim MYSQL_BOTH, vom putea accesa rezultatul atât numeric cât și asociativ.

```
$resursa = mysql_query("SELECT *
    FROM carti");
$arrRezultat =
    mysql_fetch_array($resursa);
print $arrRezultat[1]."
    ".$arrRezultat[3]."<br>";
print $arrRezultat['id_carte']."
    ".$arrRezultat['titlu'];
```

Observăm folosirea operatorului . de concatenare a stringurilor pentru a interpune spații și line break între variabilele ce urmează a fi afișate. Mai observăm că, deși mysql_num_rows ne spune că avem mai multe cărți, print_r (\$arrRezultat) sau print \$arrRezultat['titlu'] ne afișează doar primul element al array-ului. Putem afișa toate valorile array-ului folosind while:

```
$resursa = mysql_query("SELECT *
    FROM carti");
while($row=mysql_fetch_array($resursa))
{
    print $row['titlu']." "
        .$row['descriere']."<br>";
}
```

Folosind while scriptul trece prin toate valorile array-ului până la sfârșit și pentru fiecare din ele afișează valorile câmpurilor titlu și descriere. Variabila \$row este la rândul ei un array (observați că am apelat-o ca atare, \$row['titlu']) și asta pentru că \$arrRezultat este un array multidimensional.

Ce este un array multidimensional? Un array care conține la rândul lui alte array-uri. În cazul de față, \$arrRezultat este multimea array-urilor de rânduri din tabel iar fiecare \$row este un array care cuprinde valorile rândului respectiv. Spre exemplu, în pseudocod:

```
$arrRezultat = array(rândul 1,
    rândul 2, rândul 3);
$rândul1 = array(titlu de pe rândul 1,
    descriere de pe rândul 1);
$rândul2 = array(titlu de pe rândul 2,
    descriere de pe rândul 2);
$rândul3 = array(titlu de pe rândul 3,
    descriere de pe rândul 3);
```

După cum vom vedea în continuare, aceste câteva funcții ne vor fi de ajuns pentru a putea construi un site dinamic interactiv și rareori veți avea nevoie de mai mult. Vă recomand ca de câte ori sunteți în impas să consultați manualul PHP pentru a afla informații despre celelalte funcții MySQL disponibile.

Mai puțin înseamnă mai mult

Să revenim la site-ul nostru. Dacă ar fi fost să facem site-ul în HTML, pentru fiecare carte ar fi trebuit să concepem câte o pagină și pentru fiecare nouă carte ar fi trebuit să modificăm pagina domeniului căruia îi aparține. Cu PHP în schimb tot ce trebuie să facem este prima pagină, o pagină pentru domeniu și una pentru carte. Atât - trei pagini vor fi tot ce avem nevoie, indiferent de numărul cărților sau domeniilor din baza de date. În continuare vom exersa funcțiile MySQL ale PHP creând o schiță a site-ului.

Spuneam că pe prima pagină vom avea numele de domenii. Numele de domenii le obținem folosind interrogarea SELECT nume_domeniu from domenii iar pentru a le afișa, scriptul PHP pentru prima pagină va arăta ca în exemplul următor:

index.php

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<title>librăria mea</title>
</head>
<body>
<h1>librăria mea</h1>
<?
mysql_connect("localhost", "root", "");
mysql_select_db("librarie");
/* să afișăm pe prima pagină numărul de cărți din baza de date: */
$sql1 = "SELECT * FROM carti";
$resursa1 = mysql_query($sql1);
$nr = mysql_num_rows($resursa1);
print "<p>Sunt $nr carti în baza de date</p>";
?>
<h2>Domenii</h2>
<?
/* și în continuare afișăm numele de domenii: */
$sql2 = "SELECT nume_domeniu
FROM domenii";
$resursa2 = mysql_query($sql2);
while($row=mysql_fetch_array($resursa2))
{
```

O primă pagină ce conține lista domeniilor disponibile.

```
print $row['nume_domeniu'].'<br>';
?>
</body></html>
```

Scriptul și rulați acest script. E interesant, spuneți, dar nu suficient de impresionant? Ei bine, atunci să facem în așa fel încât să putem da click pe fiecare nume de domeniu și să intrăm într-o pagină unde ne sunt afișate cărțile din domeniul respectiv.

Cum facem acest lucru? Ne reamintim că am legat tabelele domenii și carti prin id_domeniu. În limbaj SQL am putea să folosim interrogarea SELECT titlu from carti, domenii where nume_domeniu='Poezii' and domenii.id_domeniu=carti.id_domeniu; pentru a afișa toate titlurile din categoria „Poezii” și la fel pentru toate celelalte categorii. Nu vom face câte o pagină pentru fiecare domeniu în parte deoarece acesta este exact scopul nostru: să scriem un singur script care să afișeze dinamic rezultatele: cărțile de poezii pentru domeniu „Poezii” și operele de teatru pentru domeniu „Teatru”. Pentru aceasta vom folosi o variabilă care să conțină numele domeniului, variabilă pe care o transmitem scriptului domeniu.php prin URL. Va trebui să modificăm în index.php linia

```
print $row['nume_domeniu'].'<br>';
?>
<?
și să transformăm numele de domeniu într-un link prin care să transmitem variabila scriptului din domeniu.php.
print '<a href="domeniu.php?
```

```
nume_domeniu='.$row['nume_domeniu'].'
">'.$row['nume_domeniu'].'</a><br>;
```

Astfel în prima pagină toate numele de domenii vor fi link-uri, ca de exemplu Poezie. Fiți atenți la disperarea ghilimelelor în cadrul scriptului, în special la modul în care am inclus ghilimele duble într-un string încadrat de ghilime simple.

Să facem în continuare scriptul care afișează titlurile disponibile în domeniu pe care am dat click. În fișierul domeniu.php vom folosi variabila trimisă prin URL pentru a obține din baza de date doar cărțile aparținând domeniului respectiv:

domeniu.php

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<title>librăria mea</title>
</head>
<body>
<?
mysql_connect("localhost", "root", "");
mysql_select_db("librarie");
$sql = "SELECT titlu FROM carti,
domenii WHERE nume_domeniu=
'".$_GET['nume_domeniu']."' AND
domenii.id_domeniu=carti.id_domeniu";
$resursa = mysql_query($sql);
$nr = mysql_num_rows($resursa);
print "<p>Sunt $nr cărți în acest
domeniu</p>";
?>
<h2>Titluri</h2>
<?
while($row=mysql_fetch_array($resursa))
```

Lista cărților din domeniul Poezii alese după id_domeniu.



Putem trimite numele domeniului prin URL.

```
{
    print $row['titlu'].'<br>';
}
?>
</body>
</html>
```

Vă spuneam în capitolul despre MySQL cum id-urile ne pot ușura munca. Acest lucru devine evident atunci când creem pagini de web dinamice deoarece sintaxa interogărilor SQL poate fi simplificată foarte mult.

Fiecare nume de domeniu din tabelul domenii are câte un id asociat și atunci, știind că domeniu Poezii are id_domeniu 1 am putea alege cărțile din domeniul respectiv folosind simplu: `SELECT * FROM carti WHERE id_domeniu=1` în loc de mai greu de intesul `SELECT titlu FROM carti, domenii WHERE nume_domeniu='Poezii' AND domenii.id_domeniu=carti.id_domeniu`. Cum? Trimînd valoarea lui id_domeniu prin URL, pentru fiecare domeniu în parte. Modificați index.php astfel:

```
$sql2 = "SELECT id_domeniu,
    nume_domeniu FROM domenii";
$resursa2 = mysql_query($sql2);
while($row =
    mysql_fetch_array($resursa2))
{
    print '<a href="domeniu.php?
        id_domeniu='.$row['id_domeniu'].'">'.
        $row['nume_domeniu'].'</a><br>';
```

Acum linkurile către pagina domenii.php vor fi de tipul `Poezii`. Să modificăm acum și interogarea

\$sql din domeniu.php și să o simplificăm folosind id_domeniu ca referință:

```
$sql = "SELECT titlu FROM carti WHERE
    id_domeniu='".$_GET['id_domeniu'];"
```

Putem, în mod similar, face o pagină care să preia id_carte din URL și să afișeze pentru cartea respectivă titlul, autorul și descrierea. Vom face acest lucru într-unul din capitolele următoare. Pentru moment însă, este suficient să înțelegeți funcțiile MySQL și modul în care se folosesc variabilele în interogările SQL.

Și chiar mai mult

Putem folosi funcția mysql_query pentru a efectua și alte operații cu baza de date: INSERT, UPDATE sau DELETE. Să facem întâi un script care ne afișează toate titlurile din tabelul carti, script de care vom avea nevoie în continuare pentru a consulta baza de date fără să fim nevoiți să intrăm în linia de comandă. Vom prezenta datele într-un tabel HTML:

toate_cartile.php

```
<?
mysql_connect("localhost", "root", "");
mysql_select_db("librarie");
$sql = "SELECT id_carte, titlu FROM
    carti";
$resursa = mysql_query($sql);
// deschidem tabelul:
print "<table border='1'>";
// scriem fiecare rezultat pe un rând:
while($row=mysql_fetch_array($resursa))
{
    print
        "<tr>
            <td>".$row['id_carte']."</td>
            <td>".$row['titlu']."</td>
        </tr>";
}
// și închidem tabelul
print "</table>";
?>
```

Rulați scriptul accesând adresa `http://localhost/toate_cartile.php` și apoi creați un nou fișier, operatii.php:

operatii.php

```
<?
mysql_connect("localhost", "root", "");
mysql_select_db("librarie");
$sql = "INSERT INTO carti(titlu)
    VALUES('Dune')";
mysql_query($sql);
?>
```

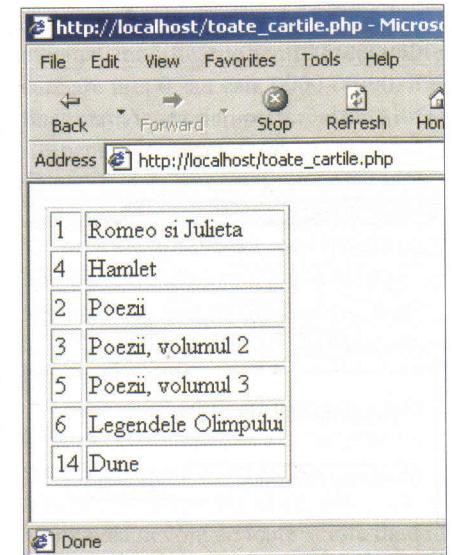


Putem trimite id-ul domeniului, pentru o orientare mai ușoară.

```
<a href="toate_cartile.php">
    Vezi modificarea!</a>
```

Rulați scriptul în browser apoi accesați `http://localhost/toate_cartile.php` pentru a vedea că noul titlu a fost introdus în tabel și i-a fost acordat automat și un id. Dacă folosiți Opera sau setarea de refresh din Internet Options din Internet Explorer „Check for newer versions of stored pages - Never”, va fi nevoie să reîncărcați pagina (refresh).

Să modificăm operații.php pentru a vedea cum folosim instrucțiunea DELETE, rescriind variabila \$sql:



Toate cărțile din baza de date.

operații.php

```
<?
mysql_connect("localhost", "root", "");
mysql_select_db("librarie");
$sql = "DELETE FROM carti WHERE
    titlu='Dune'";
mysql_query($sql);
?>
<A href="toate_cartile.php">Vezi
    modificarea!</a><br>
<A href="adaugare.html">Adaugă încă un
    titlu!</a>
```

Rulați și acest exemplu și apoi accesați http://localhost/toate_cartile.php pentru a vedea că înregistrarea a fost ștersă din tabel.

PHP, MySQL și formularele

A sosit momentul să punem cap la cap toate lucrurile învățate până acum.

Folosind formularele HTML putem interacționa cu baza de date fără să trece prin chinurile scrierii interogării SQL în linia de comandă sau în cadrul unui script PHP ori de câte ori vrem să adăugăm, să modificăm sau să ștergem vreo înregistrare.

Să facem un formular care să ne ajute în adăugarea de titluri noi:

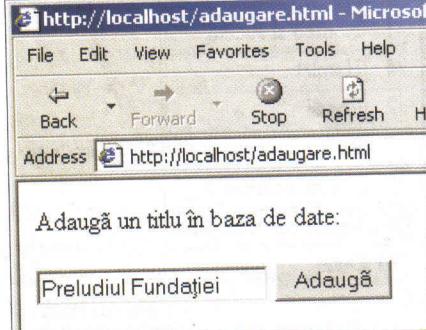
adaugare.html

```
Adaugă un titlu în baza de date:
<form action="adaugare.php"
    method="POST">
<input type="text" name="titlu">
<INPUT type="submit" value="Adaugă">
</form>
```

și iată și scriptul care va prelua valoarea din `<input type="text" name="titlu">` transmisă prin metoda POST și o va adăuga în baza de date:

adaugare.php

```
<?
mysql_connect("localhost", "root", "");
mysql_select_db("librarie");
```



Formularul în care vom introduce titlul ce urmează a fi inclus în baza de date.

```
$sql = "INSERT INTO carti(titlu)
    VALUES ('".$_POST['titlu']."'')";
mysql_query($sql);
?>
<A href="toate_cartile.php">Vezi
    modificarea!</a><br>
<A href="adaugare.html">Adaugă încă un
    titlu!</a>
```

Accesați <http://localhost/adaugare.html> și puteți adăuga oricâte titluri doriti! Vă puteți verifica oricând, accesând http://localhost/toate_cartile.php.

Să scriem acum un script cu care să ștergem titluri din baza de date și cu această ocazie vom vedea și cum folosim alte controale de formulare decât `<input type="text">` sau `<input type="submit">`.

În exemplul următor vom vedea cum afișăm titlurile într-un dropdown list pentru a putea selecta titlul care urmează să-l ștergem.

Vom folosi un script PHP pentru a afișa utilizatorului o listă cu titlurile disponibile în baza de date, urmând ca el să aleagă titlul ce urmează a fi șters.

stergere.php

```
Sterge un titlu din baza de date:
<form action="stergere_act.php"
    method="POST">
<select name="id_carte">
    <?
    mysql_connect("localhost", "root", "");
    mysql_select_db("librarie");
    $sql = "SELECT id_carte, titlu
        FROM carti ORDER BY titlu ASC";
    $resursa = mysql_query($sql);
    while($row =
        mysql_fetch_array($resursa))
    {
        print "<option
            value='".$row['id_carte']."'>
            '".$row['titlu']."'</option>";
    }
?>
</select>
<INPUT type="submit" value="Sterge">
</form>
```

În acest formular transmitem variabila cu numele `id_carte` (definită în `<select name="id_carte">`) a cărei valoare este conținută în tagul `option` (`<option value='".$row['id_carte']."'>`). Iată și scriptul care preia această variabilă și șterge rândul corespondent din baza de date:

stergere_act.php

```
<?
mysql_connect("localhost", "root", "");
```

```
mysql_select_db("librarie");
$sql = "DELETE FROM carti WHERE
    id_carte='".$_POST['id_carte']."'";
mysql_query($sql);
?>
<A href="toate_cartile.php">Vezi
    modificarea!</a><br>
<A href="stergere.php">Mai sterge
    una!</a>
```

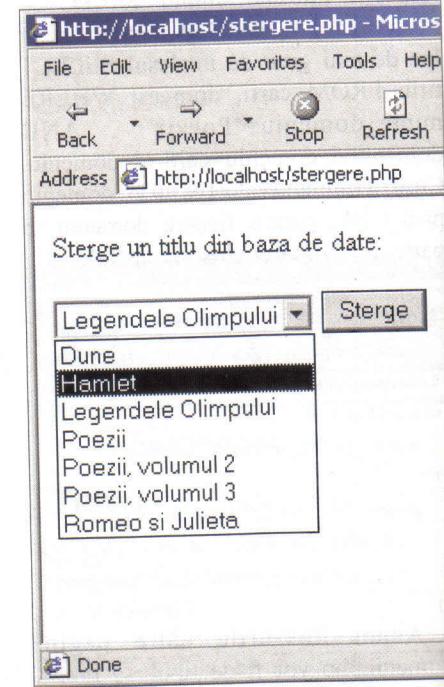
Cât de ușor este să ștergem ceva din baza de date și nu trebuie să ținem minte nici un id, nici un titlu - amândouă ne sunt servite pe tavă de către script!

Pentru a modifica datele din baza de date vom scrie trei scripturi: unul care ne afișează titlurile și ne permite să selectăm titlul pe care dorim să îl modificăm, unul care să afișeze datele într-un formular unde să le putem modifica și încă unul care să trimítă datele modificate la baza de date:

lista_carti.php

Alege una din carti și apăsa butonul Modifica:

```
<FORM action="modifica.php"
    method="POST">
<?
mysql_connect("localhost", "root", "");
mysql_select_db("librarie");
$sql = "SELECT id_carte, titlu,
    descriere FROM carti";
$resursa = mysql_query($sql);
while($row =
    mysql_fetch_array($resursa))
```



Afișarea cărților disponibile într-o listă dropdown pentru o selecție rapidă.

```

{
    print "<input type='radio'
name='id_carte'
value='".$row['id_carte']."'/>
<b>".$row['titlu']."</b><br>
<i>".$row['descriere']."</i>
<br><br>";
}
?>

<INPUT type="submit"
value="Modifică">
</form>

```

Accesați scriptul în browser pentru a vedea rezultatele. Să trecem la următorul fișier:

modifica.php

```

Modifica aceasta carte:
<FORM action="modifica_act.php"
method="POST">
?>

```

/* Luăm titlul și descrierea din nou din baza de date pentru a le afișa în controalele formularului deoarece din fișierul precedent nu am primit decât o singură variabilă, \$_POST['id_carte'], din butonul radio (<input type='radio' name='id_carte' value='".\$row['id_carte']."'/>)*/

```

mysql_connect("localhost", "root", "");
mysql_select_db("librarie");
$sql = "SELECT titlu, descriere
FROM carti WHERE
id_carte=".$_POST['id_carte'];
$resursa = mysql_query($sql);

```

```

while($row =
mysql_fetch_array($resursa))
{
    print '<input type="text"
name="titlu"
value="'.$row['titlu'].'"><br>
<textarea name="descriere">
'.$row['descriere'].
'</textarea><br>';
}

```

/* și folosind un control ascuns transmitem mai departe către modifica_act.php id-ul cărții deoarece va avea nevoie de el atunci când va opera efectiv modificarea în baza de date */

```

print '<input type="hidden"
name="id_carte"
value="'.$_POST['id_carte'].'">;
?>
<INPUT type="submit"
value="Modifică">
</form>

```

În fine, ultimul script din această serie, modifica_act.php preia variabilele \$_POST['titlu'] și \$_POST['descriere'] din formularul precedent și actualizează înregistrarea cărții a cărui id_carte are valoarea lui \$_POST['id_carte'].

La sfârșitul scriptului folosim header() pentru a redirectiona automat browserul către lista_carti.php după ce modificarea a fost efectuată:

modifica_act.php

```

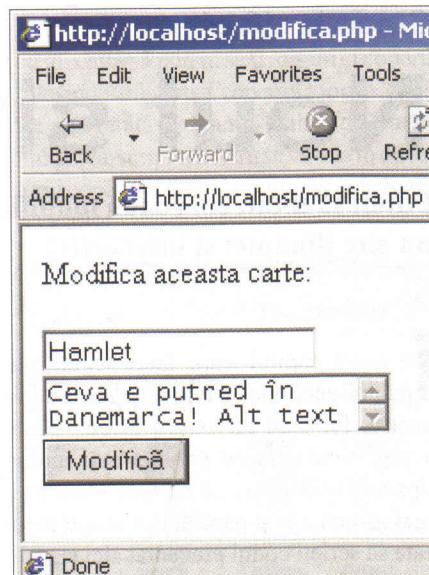
?>
mysql_connect("localhost", "root", "");
mysql_select_db("librarie");
$sql = "UPDATE carti SET
titlu='".$_POST['titlu']."',
descriere='".$_POST['descriere']."'",
WHERE id_carte=".$_POST['id_carte'];
mysql_query($sql);
header("location: lista_carti.php");

```

Accesați acum adresa http://localhost/lista_carti.php și modificați după plac orice titlu sau descriere a unei cărți! De-acum puteți spune Adio! lucrului cu baza de date din linia de comandă.

În acest capitol am văzut cum putem face pagini dinamice și interactive și cât de ușor este să creezi și să ţii „la zi” un site cu un număr uriaș de pagini folosind doar câteva scripturi.

Că începători în programare v-ai putea simți frustrați de complexitatea ghilimelelor, variabilelor sau array-urilor dar nu renunțați, aveți nevoie doar de ceva exercițiu și în curând toate vi se vor părea



Formularul cu ajutorul căruia modificăm efectiv detaliile cărții.

floare la ureche. Nu uitați: de câte ori lucrurile nu funcționează aşa cum ar trebui, verificați dacă ati pus punct și virgulă la sfârșitul propozițiilor, dacă nu ati scris greșit numele unei variabile sau funcții sau dacă ati pus toate ghilimelele la locul lor. Înainte de a trece la capitolul următor vă recomand să faceți câteva exerciții.

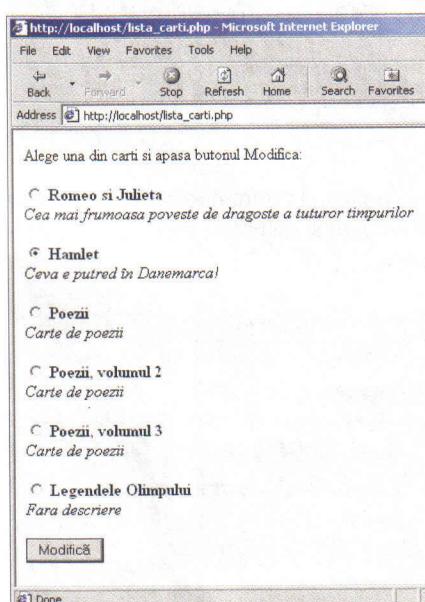
Noi avem câteva cărți în baza de date însă nici una nu are preț sau dată (de fapt valorile sunt 0 pentru preț și 0000-00-00 pentru dată).

Scrieți scripturi pentru a modifica prețul și data fiecărei cărți, la fel ca în ultimul exemplu prezentat. Nu uitați că prețul este de tip INT în baza de date și va trebui să îl scrieți ca „98500” nu „98,500”, „98500 lei” sau „98,500”. La fel, data este în format yyyy-mm-dd (ex: 2003-02-23).

După ce ati scris scripturile, rulați și efectuați modificările pentru fiecare carte care are preț 0 sau data 0000-00-00. Fiți imaginativi și scrieți prețuri și date diferite pentru fiecare carte. Vom avea nevoie de aceste date în capitolele următoare.

Înainte de a continua, verificați dacă toate cărțile din tabelul carti aparțin unui domeniu (id_domeniu nu este 0 și se regăsește și în tabelul domenii) și au autor (id_autor nu este 0 și se regăsește și în tabelul autorii).

Dacă nu, modificați înregistrările (în linia de comandă sau cu ajutorul unui script PHP) astfel încât toate cărțile să aibă un id_domeniu și id_autor valid. Vom avea nevoie de toate aceste date în cele ce urmează.



Formularul cu ajutorul căruia putem alege cartea ale cărei date dorim să le modificăm.

Crearea siteului

Noul site pas cu pas

Puteți să uitați corvoada modificărilor și upload-urilor zilnice: cunoașteți suficient încât să puteți crea un site dinamic și interactiv!

In acest capitol vom face pagină cu pagină secțiunea navigabilă a site-ului nostru, folosind tot ce am învățat până acum. Vom afla pe parcurs mai multe tips and tricks care să ne facă munca și mai ușoară. Ca și până acum, sfatul meu este să scrieți codul prezentat aici și să îl testați voi însivă, multe concepte pot părea greu de digerat pe hârtie și foarte ușor de înțeles în practică.

Vom face întâi curățenie în Document root (c:\Program files\Apache Group\Apache\htodcs). Ștergeți sau mutați fișierele deja existente acolo pentru a nu vă încurca mai departe. Apoi asigurați-vă că serverul MySQL este pornit și deschideți editorul PHP preferat.

Prima pagină

O primă pagină atractivă este vitală pentru succesul unui site. Menirea ei este să ofere utilizatorului motivele pentru a continua să vadă mai mult și eventual să fie imediat interesat să cumpere ce avem de oferit. Vitrina librăriei noastre virtuale va prezenta utilizatorului atât domeniile de carte disponibile cât și cele mai noi cărți adăugate și precum și cele mai populare. Vom mai avea și o căsuță pentru căutare pentru cei grăbiți sau gata hotărâți pentru ca aceștia să poată găsi rapid ceea ce îi interesează.

Prima pagină este de fapt o combinație din mai multe pagini. Unele elemente sunt prezente pe toate paginile din acest site, ca de exemplu notița de copyright de la sfârșit, instrucțiunile de conectare la baza de date sau prima parte în care specificăm titlul, setul de caractere folosit și stilul CSS.

PHP ne oferă funcția include cu care le putem include în cadrul scriptului și să le refolosim în orice alt script în care avem nevoie de ele fără să trebuiască să le scriem din nou.

Structura fiecărei pagini din site-ul nostru, indiferent dacă este prima pagină, pagina de domeniu sau pagina cu detalii despre carte, va arăta astfel:

conectarea la baza de date

partea de sus a paginii

meniu

conținutul paginii

partea de jos a paginii

conectare.php

```
mysql_connect("localhost", "root", "");  
mysql_select_db("librarie");
```

Fisierul *page_top.php* va conține tagurile clasice cu care se începe un fișier HTML, setul de caractere și titlul paginii, logo-ul și imediat după, deschide tabelul ce urmează să cuprindă meniul și conținutul paginii:

page_top.php

```
<html>  
<head>  
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">  
<title>libraria mea</title>  
<style type="text/css">  
body, p, td {font-family: Verdana, Arial, sans-serif; font-size: 12px;}  
h1 {font-family: Times New Roman, Times, serif; font-size: 18px; font-weight: bold; color: #336699; font-style: italic;}  
.titlu {font-family: Verdana, Arial, sans-serif; font-size: 14px; font-weight: bold; color: #0066CC;}  
</style>  
</head>  
<body bgcolor="#ffffff">  
  
<table>  
  <tr>
```

conectare.php

page_top.php

meniu.php

conținutul paginii

page_bottom.php

În *page_bottom.php* scriem notița de copyright și închidem toate tagurile deschise.

page_bottom.php

```
</tr>  
</table>  
<p align="center"><a href="http://www.vogelburda.ro">&copy; Vogel Burda Communications</a></p>  
</body>  
</html>
```

Fisierul *conectare.php* va conține cele două instrucțiuni, *mysql_connect* și *mysql_select_db* pe care le vom folosi pe fiecare pagină a site-ului.

Pagina *meniu.php* va conține toate domeniile de carte disponibile astfel încât ele să fie accesibile în orice moment de

pe orice pagină a site-ului, precum și caseta de căutare. În acest moment suntem conectați la baza de date, deoarece fișierul meniu.php nu îl vom accesa direct ci îl includem după ce am inclus fișierul conectare.php. Putem deci să scriem direct scriptul pentru afișarea numelor de domenii, în ordine crescătoare (Aventuri, Biografii, Clasici, etc.).

meniu.php

```
<td valign="top" width="125">
<div style="width:120px; background-color:#F9F1E7; padding:4px; border:solid #632415 1px">
<b>Alege domeniul</b><HR size="1">
<?
$ssql = "SELECT * FROM domenii ORDER BY nume_domeniu ASC";
$resursa = mysql_query($ssql);
while($row =
mysql_fetch_array($resursa))
{
print '<a href="domeniu.php?id_domeniu='.$row['id_domeniu'].'">'.$row['nume_domeniu'].'
</a><br>';
}
?>
</div>
<br>
<div style="width:120px; background-color:#F9F1E7; padding:4px; border:
```

```
solid #632415 1px">
<form action="cautare.php" method="GET">
<b>Căutare</b><br>
<INPUT type="text" name="cuvant" size="12"><br>
<INPUT type="submit" value="Căută">
</form>
</div>
</td>
```

Și, în final, să creem un fișier index.php care să le cuprindă pe toate. În el vom adăuga conținut din baza de date astfel încât să avem o primă pagină dinamică. Priviți pentru un moment cum va arăta pagina noastră după ce vom termina de scris codul. Îmi și imaginez ce veți spune: „Are imagini și până acum nu a spus nimănui nimic despre imagini!”. Puteți transfera fișiere pe server folosind formulare și PHP însă, cum subiectul depășește tema propusă, vă sugerez să consultați manualul PHP de pe CD dacă dorîți să aflați mai multe. Mai simplu, putem crea „manual” câte o imagine în format JPEG de 75 de pixeli lățime și 100 de pixeli înălțime pentru fiecare copertă a cărților din baza de date și să îi dăm numele 1.jpg pentru coperta cărții cu id_carte=1, 2.jpg pentru coperta cărții cu id_carte=2 și aşa mai departe. Toate aceste imagini le vom pune într-un director numit coperte în document root.

Funcția include

Mai ușor decât Copy&Paste

Cu ajutorul funcției include putem refolosi bucăți de cod fără să fie nevoie să le scriem ori de câte ori avem nevoie de ele, indiferent dacă este cod PHP precum instrucțiunile de conectare la baza de date sau HTML precum notiță de copyright de pe fiecare pagină a site-ului.

Trebuie doar să scriem părțile refolosibile în fișiere separate pe care să le accesăm atunci când avem nevoie de ele. Iată cum includem trei fișiere în altul:

top.html

```
<html>
<body bgcolor="#FFFFCC">
<h1>titlul paginii</h1>
```

copyright.html

```
<p><a href="http://www.vogelburda.ro">&copy; Vogel Burda Communications</a></p>
```

variabile.php

```
<?
print "un text conținut în fișierul variabile.php";
$variabila = "și o variabilă din
```

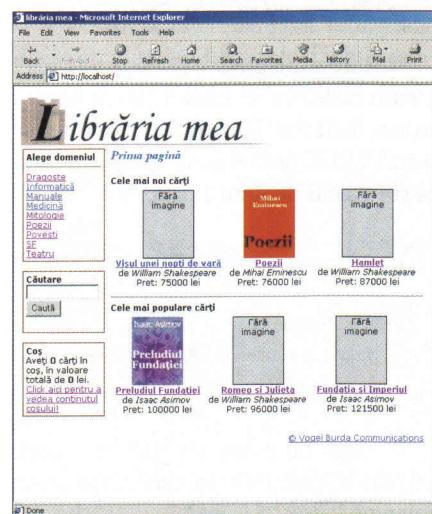
```
fișierul variabile.php";
?>
```

Și iată și fișierul care le folosește pe toate acestea:

```
test1.php
<?
include("top.html");
?>
<p>Un text oarecare!</p>
<?
include("variabile.php");
print $variabila;
include("copyright.html");
?>
```

Observăm că fișierele care conțin cod PHP trebuie să conțină tagurile <? și ?>, aceasta pentru că parserul iese din modul PHP la începutul fișierului inclus și trece în mod HTML, reluând modul PHP atunci când termină de evaluat fișierul respectiv. Variabilele, funcțiile și resursele din fișierele incluse sunt disponibile în cadrul scriptului după ce au fost incluse. Dacă setarea URL_fopen_wrappers este activată în php.ini puteți include chiar și fișiere aflate pe alte servere: include("http://www.chip.ro").

Astfel, când afișăm informația despre carte cu id_carte=3 putem să îi afișăm și coperta folosind . Pentru mai multă siguranță, înainte de a afișa imaginea, scriptul nostru se va uita în directorul coperte și doar dacă va găsi imaginea, o va afișa.



Prima pagină conține cele mai importante elemente.

index.php

```
<?
include("conectare.php");
include("page_top.php");
include("meniu.php");
?>
<td valign="top">
<h1>Prima pagină</h1>
<b>Cele mai noi cărți</b>
<table cellpadding="5">
<tr>
```

/* În următoarea interogare selectăm informațiile despre cele mai noi trei cărți și le afișăm pe fiecare într-o celula de tabel: */

```
$ssql = "SELECT id_carte, titlu,
nume_autor, pret FROM carti, autori
WHERE carti.id_autor=autori.id_autor
ORDER BY data DESC LIMIT 0,3";
$resursa = mysql_query($ssql);
while($row =
mysql_fetch_array($resursa))
{
/* deschidem celula tabelului
HTML */
print '<td align="center">';
/* Să punem și imaginea copertei.
```

Este posibil ca nu pentru toate cărțile să avem o imagine, aşa că vom face după cum urmează: dacă avem imagine pentru copertă, o afișăm. Dacă nu, afișăm un

layer DIV în care scriem simplu „Fără imagine”. Vom compune întâi adresa imaginii, știind că aceasta se află în directorul coperte, are același nume ca id_carte și extensia jpg: */

```
$adresaImagine = "coperte"
.$row['id_carte'].".jpg";
```

/* Adresa imaginii va fi "coperte/1.jpg" pentru cartea cu id_carte 1. Dacă fișierul există, îl afișăm. Funcția file_exists returnează TRUE dacă fișierul specificat există și FALSE dacă nu există. */

```
if(file_exists($adresaImagine))
{
    print '<br>';
}

/* Dacă nu există fișierul specificat,
afișăm layerul DIV în care scrie „Fără
imagină”: */

else
{
    print '<div style="width:75px;
    height:100px; border: 1px black
    solid; background-color:#cccccc">
    Fără imagine</div>';
}

/* Să continuăm cu afișarea restului de
informații, titlul, numele autorului, prețul: */


```

```
print '<b><a href="carte.php?
    id_carte='.$row['id_carte'].'">'
    .$row['titlu']..'</a></b><br> de
    <i>'.$row['nume_autor']..'</i><br>
    Pret: '.$row['pret'].' lei
```

/* Închidem celula <td> cu care am început structura while. While va afișa toate cele trei cărți selectate în interogarea SQL și pentru fiecare din ele va repeta pașii de mai sus. */

```
</td>;
}

/* Am terminat cu cele mai noi cărți.
Închidem tabelul, tragem o linie de separare după care vom scrie același cod pentru cele mai populare cărți, modificând doar interogarea SQL. */

?>
</tr>
</table>
<hr>
<b>Cele mai populare cărți</b>
<table cellpadding="5">
    <tr>
        <?
        /* Care sunt cele mai populare cărți?
```

Cele mai vândute, desigur. Aflăm care sunt cele mai vândute cărți, consultând tabelul vânzări, cu interogarea următoare. Dacă nu aveți nici un fel de date în tabelul vânzări, interogarea va returna 0 rânduri (nici o carte vândută) și nu va apărea nimic pe prima pagină în secțiunea „Cele mai populare cărți”. Introduceți câteva date în acest tabel. */

```
$sqlVanzari = "SELECT id_carte,
    sum(nr_buc) AS bucatiVandute FROM
    vanzari GROUP BY id_carte ORDER BY
    bucatiVandute DESC LIMIT 0,3";
```

/* Notă explicativă privind această interogare: din tabelul vânzări ne interesează două coloane: id_carte și nr_buc (cărți vândute). Dacă grupăm id-urile, putem afla numărul de total de bucați vândute din fiecare carte folosind funcția MySQL sum(nr_buc). Id-urile le ordonăm apoi crescător în funcție de numărul total de bucați vândute din fiecare (ORDER BY bucatiVandute). bucatiVandute este de fapt un alias pentru coloana creată ad-hoc, sum(nr_buc).

Definirea unui alias se poate face folosind AS ca în SELECT sum(nr_buc) AS bucatiVandute. Am definit această coloană creată instant ca alias deoarece nu am fi putut face ordonarea direct ORDER BY sum(bucatiVandute). Puteți rula această interogare în linia de comandă pentru a vedea rezultatul aşa cum îl oferă MySQL. Să continuăm cu scriptul PHP: */

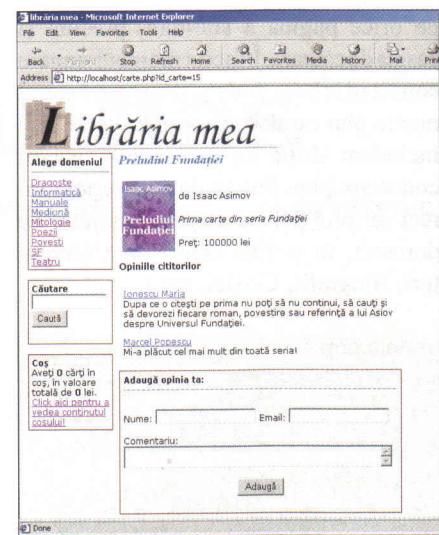
```
$resursaVanzari=mysql_query($sqlVanzari);
```

/* Valorile din acest query sunt trei id_carte din tabelul vânzări care corespund celor mai trei vândute cărți și numărul total de bucați vândute din fiecare. Vom folosi aceste id-uri pentru a interoga, cu fiecare din ele, baza de date și a afla titlul, autorul și prețul fiecareia din ele: */

```
while($rowVanzari =
    mysql_fetch_array($resursaVanzari))
{
    $sqlCarte = "SELECT titlu,
        nume_autor, pret FROM carti,
        autori WHERE carti.id_autor=
        autori.id_autor AND id_carte=".
        $rowVanzari['id_carte'];
    $resursaCarte=mysql_query($sqlCarte);
```

/* Acum avem toate datele care ne interesează: id_carte (din interogarea \$sqlVanzari), titlul, numele autorului și numărul de bucați vândute.

Să le afișăm: */



Comentariile utilizatorilor la această carte.

```
while($rowCarte =
    mysql_fetch_array($resursaCarte))
{
    print '<td align="center">';
    $adresaImagine = "coperte/
    '.$rowVanzari['id_carte'].'.jpg";
    if(file_exists($adresaImagine))
    {
        print '<br>';
    }
    else
    {
        print '<div style="width:75px;
        height:100px; border: 1px black
        solid; background-color:#cccccc">
        Fără imagine</div>';
    }

    print '<b><a href="carte.php?id_carte='.
    $rowVanzari['id_carte'].'">'.
    $rowCarte['titlu']..'</a></b><br> de
    <i>'.$rowCarte['nume_autor']..'</i><br>
    <br>Pret: '.$rowCarte['pret'].' lei
    </td>';
}
?>
</tr>
</table>
</td>
</tr>
</table>
<?

include("page_bottom.php");
?>
```

Acesta a fost index.php, prima pagină a site-ului nostru. Întotdeauna informațiile de pe ea vor fi la zi. Puteți adăuga o

carte nouă în baza de date pentru a vedea că apare imediat apoi și pe prima pagină (nu uitați să menționați și data deoarece acesta este criteriu de ordonare!).

Domenii

A doua pagină pe care o vom scrie este cea care se ocupă de domenii. Am și scris deja, în capitolul anterior, un mic script care să ne afișeze rapid titlurile cărților dintr-un domeniu ales. Același lucru vom face în continuare, ceva mai elaborat. În pagina domeniu.php vom afișa cărțile ce aparțin domeniului selectat, pozele copertelor, numele autorilor și prețul. În acest script doar interogarea SQL este mai complicată deoarece „leagă” trei tabele (cărți, domenii, autori) dar restul ar trebui să vă pară deja floare la ureche! Iată și un mic tip înainte de a începe. Atunci când doriți să afișați valoarea unei variabile în cadrul codului HTML, în loc să scrieți:

```
cod HTML
<?
print $numeVariabila;
?>
cod HTML
```

puteți apela la mult mai simplul:

```
cod HTML <?=$numeVariabila?> cod HTML
```

Să trecem la treabă și să folosim în următorul script această nouă facilitate:

```
domeniu.php
<?
include("conectare.php");
include("page_top.php");
include("meniu.php");
$id_domeniu = $_GET['id_domeniu'];
$sqlNumeDomeniu = "SELECT
    nume_domeniu FROM domenii WHERE
    id_domeniu=".$id_domeniu;
$resursaNumeDomeniu =
    mysql_query($sqlNumeDomeniu);
$numeDomeniu = mysql_result
    ($resursaNumeDomeniu,0,"nume_domeniu");
?>
<td valign="top">
<h1>Domeniu: <?=$numeDomeniu?></h1>
<b>Cărți în domeniul
<u><i><?=$numeDomeniu?></i></u></b>
<table cellpadding="5">
<?
$sql = "SELECT id_carte, titlu,
    descriere, pret, nume_autor FROM
    carti, autori, domenii WHERE
    carti.id_domeniu=domenii.id_domeniu
```

```
    AND carti.id_autor=autori.id_autor
    AND domenii.id_domeniu=".$id_domeniu;
$resursa = mysql_query($sql);
while($row=mysql_fetch_array($resursa))
{
?>
<tr>
<td align="center">
<?
$adresaImagine = "coperte"
    .$row['id_carte'].".jpg";
if(file_exists($adresaImagine))
{
    print '<img src=
        "'.$adresaImagine.'" width="75"
        height="100"><br>';
}
else
{
    print '<div style="width:75px;
        height:100px; border: 1px black
        solid; background-color:#cccccc">
        Fără imagine</div>';
}
?>
</td>
<td>
<b><a href="carte.php?
    id_carte=<?=$row['id_carte']?>">
    <?=$row['titlu']?></a></b><br>
    <i>de <?=$row['nume_autor']?></i>
    <br>Preț: <?=$row['pret']?> lei
    </td>
</tr>
<?
}
?>
</table>
</td>
<?
include("page_bottom.php");
?>
```

```
include("meniu.php");
$id_carte = $_GET['id_carte'];
$sql = "SELECT titlu, nume_autor,
    descriere, pret FROM carti, autori
    WHERE id_carte=".$id_carte." AND
    carti.id_autor=autori.id_autor";
$resursa = mysql_query($sql);
```

// Deoarece interogarea nu returnează decât un rând, nu vom folosi while pentru a itera prin toate elementele array-ului ci le vom accesa direct.

```
$row = mysql_fetch_array($resursa);
?>
<td valign="top">
<table>
<tr>
<td valign="top">
<?
$adresaImagine = "coperte/"
    .$id_carte.".jpg";
/* Dacă avem imagine pentru copertă, o afișăm, iar dacă nu avem, nu afișăm nimic */
if(file_exists($adresaImagine))
{
    *
    print '<img src=
        "'.$adresaImagine.'" width="75"
        height="100" hspace="10"><br>';
}
?>
</td>
<td valign="top">
<h1><?=$row['titlu']?></h1>
<i>de <b><?=$row['nume_autor']?>
    </b></i>
<p><i><?=$row['descriere']?>
    </i></p>
    <p>Preț: <?=$row['pret']?> lei</p>
    </td>
</tr>
</table>
<p><b>Opiniile cititorilor</b></p>
?>
$sqlComentarii = "SELECT * FROM
    comentarii WHERE id_carte=".$id_carte;
$resursaComentarii =
    mysql_query($sqlComentarii);
while($row =
    mysql_fetch_array($resursaComentarii))
{
    print '<div style="width:400px;
        border:1px solid #ffffff; back-
        ground-color:#F9F1E7;
        padding:5px"><a
        href="mailto:'.$row['adresa_email'].'">
        '.$row['nume_utilizator'].'
        <br>'.$row['comentariu'].'
        </div> ';
}
```

```
carte.php
<?
include("conectare.php");
include("page_top.php");
```

```

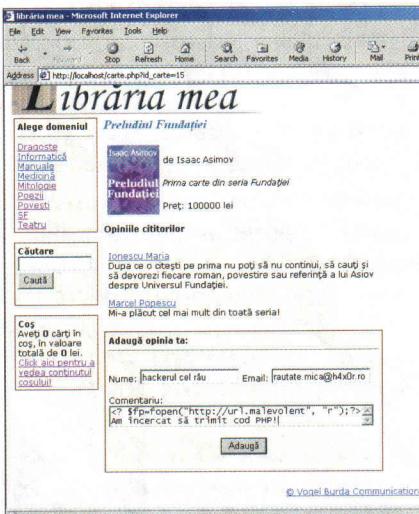
?>
<br>
<div style="width:400px; border:1px solid #632415; background-color:#F9F1E7; padding:5px;">
<b>Adaugă opinia ta:</b>
<hr size="1">
<form action="adauga_comentariu.php" method="POST">
    Nume: <input type="text" name="nume_utilizator">
    Email: <input type="text" name="adresa_email"><br><br>
    Comentariu:<br>
        <textarea name="comentariu" cols="45"></textarea><br><br>
    <input type="hidden" name="id_carte" value="<?=$id_carte?>">
    <center><input type="submit" value="Adaugă"></center>
</form>
</div>
</td>
<?
include("page_bottom.php");
?>

```

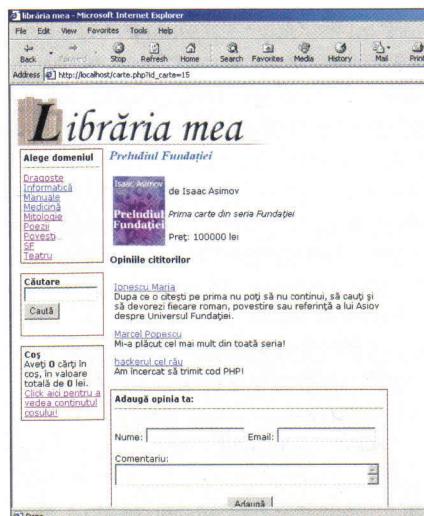
În pagina fiecărei cărți avem un formular cu ajutorul căruia utilizatorii să-și poată împărtăși impresiile precum și lista impresiilor deja adăugate. Iată fișierul adauga_comentariu.php care prelucrează informația trimisă prin formular și o introduce în baza de date:

adauga_comentariu.php

/* Verifică dacă toate câmpurile formularului de comentarii au fost complete și dacă oricare din ele a fost omis, afișează mesajul „Trebuie să completezi toate câmpurile!” și întrerupe execuția



Putem preveni multe probleme dacă folosim strip_tags pentru a elimina tagurile din inputul utilizatorilor.



Codul HTML și PHP a fost îndepărtat. Comentariul apare ca text simplu!

scriptului */

```

<?
if($_POST['nume_utilizator'] == ""
    || $_POST['adresa_email'] == ""
    || $_POST['comentariu'] == "") {
    print "Trebuie să completezi
          toate câmpurile!";
    exit;
}

```

/* Dacă execuția scriptului a ajuns până aici (adică a trecut cu succes de condiția de mai sus) înseamnă că toate câmpurile au fost completate.

Așadar, se conectează la baza de date, prelucrează informațiile transmise din formular și le introduce în baza de date:*/

```
include("conectare.php");
```

/* Folosim din motive de securitate funcția strip_tags pentru a elibera tagurile HTML și PHP din toate stringurile trimise de utilizator.

E bine să folosim strip_tags pentru a „curăța” inputul utilizatorilor de orice cod potential răuvoitor.*/

```

$numeFaraTags =
    strip_tags($_POST['nume_utilizator']);
$emailFaraTags =
    strip_tags($_POST['adresa_email']);
$comentariuFaraTags =
    strip_tags($_POST['comentariu']);
$sql = "INSERT INTO comentarii
        (id_carte, nume_utilizator,
         adresa_email, comentariu)
        VALUES ('$_POST['id_carte']',
                '$numeFaraTags', '$emailFaraTags',
                '$comentariuFaraTags')";
mysql_query($sql);

```

/* Apoi redirectionăm utilizatorul către pagina cărții la care a adăugat un comentariu:*/

```

$inapoi = "carte.php?id_carte=".
$_POST['id_carte'];
header("location: $inapoi");
?>
```

Coșul de cumpărături

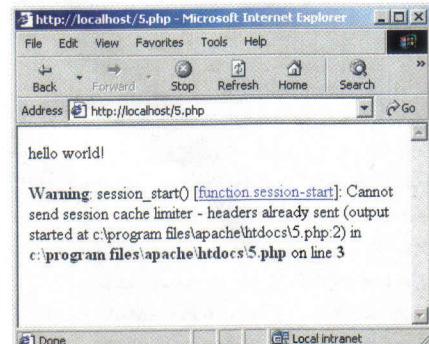
Nu am uitat că menirea principală a acestui site nu este doar de a prezenta cărți și a oferi posibilitatea ca utilizatorii să-și împărtășească impresiile despre ea ci și să cumpere! Vom face o singură pagină pentru coșul de cumpărături, pagină cu ajutorul căreia vom putea adăuga, modifica sau scoate cărți din coș și astă în mai puțin de 100 de linii de cod. Dar înainte de a începe ne întrebăm, pe bună dreptate: dacă vizitatorul adaugă o carte în coș dar apoi dorește să cumpere altă carte, unde se păstrează informațiile despre prima carte vândută? Nu se pierd odată ce a ieșit din pagina cos.php?

Am putea oare să le păstrăm în baza de date? Am putea dar pentru asta ar trebui să corelăm informația din baza de date cu utilizatorul și ar fi destul de complicat.

Am putea să trimitem informațiile despre cartea cumpărată mai departe prin URL? Si aşa ar fi prea complicat. Din fericire, PHP ne oferă o metodă foarte ușoară și la îndemână de a păstra informațiile, pe tot timpul cât este vizitat site-ul: sesiunile.

Sesiuni

Sesiunile sunt mijlocul prin care putem păstra o serie de informații în memorie. Astfel, cumpărătorul nostru poate adăuga o carte în coș și apoi să continue să viziteze site-ul. Informațiile despre cartea sau cărțile deja adăugate în coș se vor



Sesiunea nu poate fi pornită după ce am trimis deja un output către browser.

păstra în memorie și noi le putem accesa doar atunci când este nevoie. Sesiunile expira după 3 ore (puteți lăsa browserul deschis peste noapte și dimineață să reîncărcați o pagină care afișează niște variabile de sesiune pentru a vedea că nu sunt acolo) sau după ce toate instanțele browserului sunt închise. Într-o sesiune datele pot fi salvate într-o variabilă de tip array, numită `$_SESSION`. Înainte de a folosi această variabilă pentru a stoca informații trebuie să apelăm funcția predefinită `session_start()`. `session_start` pornește o sesiune dacă nu există niciuna sau o reinicializează/continuă pe cea existentă, dacă aceasta a fost deja pornită.

Informațiile din sesiune, spre deosebire de cookie-uri, sunt stocate pe server directorul pentru fișiere temporare. Pe unele sisteme Windows, dacă directorul temporar nu este setat cum trebuie, va trebui să deschideți `php.ini` și să setați chiar voi directorul temporar unde să fie păstrate sesiunile. Pentru a vedea dacă configurația dvs. este corectă, creați un nou fișier `sesiune.php` în care scrieți:

```
<?
session_start();
?>
```

Rulați-l în browser. Dacă primiți un mesaj de eroare (Warning!), va trebui să setați în `php.ini` directorul temporar. Modificați linia

```
session.save_path=/tmp
```

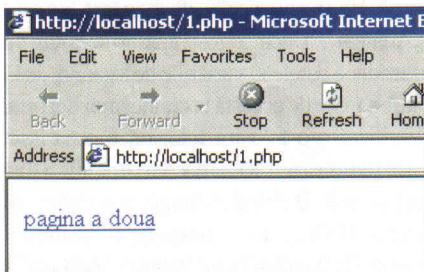
în

```
session.save_path=c:\temp\
```

Poate fi necesar să reporniți serviciul Apache pentru ca modificările să fie aplicate. Accesați acum pagina. Dacă este goală și nu ați primit nici un mesaj de eroare, configurația dvs. este corectă.

De ajuns cu „limba de lemn”, să vedem sesiunile la lucru.

În pagina `1.php` inițializăm sesiunea și



Pornim sesiunea și adăugăm câteva date în ea, nu afișăm nimic utilizatorului

stocăm două valori:

```
1.php
<?
session_start();
$_SESSION['titlu'] = "Dune";
$_SESSION['autor'] = "Frank Herbert";
?>
<a href="2.php">pagina a doua</a>
```

```
2.php
<?
session_start();
$_SESSION['altaVariabila'] = "un text
oarecare";
?>
<a href="3.php">pagina a treia</a>
```

Reinițializăm sesiunea pornită în pagina anterioară și mai adăugăm date în ea.

În pagina `2.php` continuăm sesiunea folosind `session_start` după care mai stocăm o variabilă:

```
2.php
<?
session_start();
$_SESSION['altaVariabila'] = "un text
oarecare";
?>
<a href="3.php">pagina a treia</a>
```

```
3.php
<?
session_start();
print $_SESSION['titlu']."<br>";
print $_SESSION['autor']."<br>";
print $_SESSION['altaVariabila']."<br>";
?>
<a href="4.php">pagina a patra</a>
```

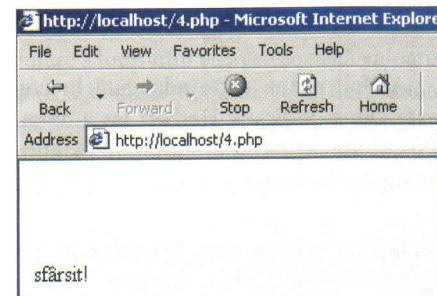
Afișăm datele stocate în sesiune în paginile anterioare.

În pagina `3.php` folosim din nou `session_start` după care afișăm toate datele stocate până acum în această sesiune:

```
3.php
<?
session_start();
print $_SESSION['titlu']."<br>";
print $_SESSION['autor']."<br>";
print $_SESSION['altaVariabila']."<br>";
?>
<a href="4.php">pagina a patra</a>
```

Toate variabilele salvate în această sesiune au fost afișate! Iată și `4.php`:

```
4.php
<?
print $_SESSION['titlu']."<br>";
```



Eroare: datele stocate în sesiune nu sunt disponibile deoarece nu am reinicializat sesiunea.

```
print $_SESSION['autor']."<br>";
print $_SESSION['altaVariabila']."<br>";
?>
```

sfârșit!

De ce nu vedem valorile și pe a patra pagină? Pentru că nu am continuat sesiunea cu `session_start`. Întotdeauna trebuie să reinicializăm sesiunea folosind `session_start` înainte de a adăuga sau manipula valori în sesiune! Pe lângă aceasta, mai există o restricție: `session_start` trebuie folosit întotdeauna înainte de a trimite date către browser altfel primim un mesaj de eroare:

5.php

```
<?
print "hello world!<br>";
session_start();
?>
```

Această regulă este valabilă chiar și pentru tag-ul `<html>` care nu produce un efect vizibil în browser, exemplul următor va da aceeași eroare:

6.php

```
<html>
<?
session_start();
?>
```

Țineți minte, întotdeauna rulați `session_start` înainte de a trimite orice output către browser!

Sesiunile ne pot ajuta să păstrăm în memorie informațiile despre cărțile deja cumpărate de către un vizitator. Cum salvăm totuși mai mult decât o carte? Folosind variabilele de sesiune pentru a stoca array-uri. Iată, de exemplu, cum am putea stoca variabilele corespunzătoare fiecărei cărți, ca un array multidimensional:

```
$titlu[1] = "Dune";
$autor[1] = "Frank Herbert";
$titlu[2] = "Poezii";
```

```
$autor[2] = "Mihai Eminescu";
```

Astfel, prima carte adăugată în coș este Dune de Frank Herbert iar a doua, Poezii de Mihai Eminescu. Iată cum adăugăm încă una:

```
$titlu[] = "Legendele Olimpului";
$autor[] = "Alexandru Mitru";
```

N-am specificat nici un număr deoarece folosind [] datele sunt adăugate automat la sfârșitul array-ului și astfel vom avea valoarea „Legendele Olimpului” pentru \$titlu[3].

Operatorul [] pentru adăugarea de valori într-un array ne va folosi pentru a nu ține tot timpul minte ce număr se află între parantezele []. Putem să adăugăm valori într-un array începând de la 0, doar folosind [], nespecificând nici un număr. Iată spre exemplu cum am putea accesa valorile array-ului specificând valoarea indexului numeric:

7.php

```
$titlu[] = "Dune";
$titlu[] = "Poezii";
$titlu[] = "Legendele Olimpului";
print $titlu[0]; // va afișa Dune
print $titlu[1]; // va afișa Poezii
print $titlu[2]; // va afișa Legendele
Olimpului
```

Și iată cum accesăm toate valorile array-ului, folosind for:

```
$titlu[] = "Dune";
$titlu[] = "Poezii";
$titlu[] = "Legendele Olimpului";
$nrElementeInArray = count($titlu);
for($i = 0; $i < $nrElementeInArray;
    $i++)
{
    print $titlu[$i]."  
";
```

Astfel, pentru \$i=0, va fi afișată valoarea lui \$titlu[0], pentru \$i = 1 valoarea lui \$titlu[1] și aşa mai departe până când când se ajunge la limita reprezentată de numărul de elemente din array. La fel vom folosi array-uri pentru a salva în variabile de sesiune informațiile despre cărțile cumpărate. Să facem întâi un mic exercițiu. Să trecem prin 3 fișiere, presupunând că în fiecare se cumpără câte o carte:

1.php

```
<?
session_start();
```

```
$_SESSION['titlu'][] = "Dune";
$_SESSION['autor'][] = "Frank Herbert";
$_SESSION['pret'][] = 100000;
$_SESSION['nr_buc'][] = 1;
?>
<a href="2.php">pagina a doua</a>
```

2.php

```
<?
session_start();
$_SESSION['titlu'][] = "Poezii";
$_SESSION['autor'][] = "Mihai Eminescu";
$_SESSION['pret'][] = 100000;
$_SESSION['nr_buc'][] = 1;
?>
<a href="3.php">pagina a treia</a>
```

3.php

```
<?
session_start();
$_SESSION['titlu'][] = "Manual de
geografie";
$_SESSION['autor'][] = "Popescu Ion";
$_SESSION['pret'][] = 50000;
$_SESSION['nr_buc'][] = 30;
```

/* avem până acum trei cărți cumpărate, să și trecem la afișarea lor folosind for, folosind câte un rând de tabel <tr> pentru fiecare carte: */

```
print '<table border="1">';
```

/* ne ajunge să știm câte titluri sunt pentru a obține întâi numărul de cărți cumpărate (numărul de rânduri din tabel, numărul de loop-uri din for)*/

```
$nrCarti = count($_SESSION['titlu']);
for($i = 0; $i < $nrCarti; $i++)
{
    print '<tr>';
    print '<td>'.
        $_SESSION['titlu'][$i].'</td>';
    print '<td>'.
        $_SESSION['autor'][$i].'</td>';
    print '<td>'.
        $_SESSION['pret'][$i].'</td>';
    print '<td>'.
        $_SESSION['nr_buc'][$i].'</td>';
    /* calculăm și cât este totalul pentru această carte, pentru Poezii va fi 1 buc x 100000 lei = 100000 lei în timp ce pentru Manualul de geografie va fi 30 buc x 50000 lei = 1500000 lei*/
}
```

```
$total = $_SESSION['nr_buc'][$i] *
$_SESSION['pret'][$i];
print '<td>'.$total.'</td>';
print '</tr>';
```

/* înainte de a încheia loop-ul, vom adăuga această valoare la totalul general. Variabila \$totalGeneral va fi inițializată la prima rulare a loop-ului și va avea valoarea totalului primei cărți. La a doua

Dune	Frank Herbert	100000	1	100000
Poezii	Mihai Eminescu	100000	1	100000
Manual de geografie	Popescu Ion	50000	30	1500000
Total General: 1700000				

Datele din array, afișate sub formă de tabel.

rulare a loop-ului va avea această valoare plus valoarea totalului celei de-a doua cărți și aşa mai departe până când se termină loop-ul. Atunci valoarea sa va fi suma totalurilor. */

```
$totalGeneral = $totalGeneral +
    $total;
}
print '</table>';
print 'Total General: '.$totalGeneral;
?>
```

Să vedem acum cum am putea folosi un singur fișier pentru a adăuga cărți în tabel. Vom profita de faptul că orice cod PHP se execută pe server și abia apoi output-ul său este trimis către browser. Să facem un fișier test.php în care vom avea un formular al cărui action să fie chiar test.php. Înainte de a afișa formularul, însă, vom adăuga în variabile de sesiune datele trimise chiar prin formular. Dacă aceasta este prima accesare a scriptului, nici un fel de date nu vor fi fost trimise, și vom trece peste codul de adăugare în variabile de sesiune. Dacă apoi completăm câmpurile formularului și apăsăm butonul „Adaugă”, scriptul de la începutul paginii va recunoaște că sunt date trimise, le va salva în variabile de sesiune și apoi le va afișa într-un tabel, după formular.

test.php

```
<?
// Pornim sesiunea
session_start();
/* Verificăm dacă a fost apăsat butonul „Adaugă”. Dacă a fost apăsat scriptul va introduce datele în variabile de sesiune, dacă nu, va trece de această secțiune, mai departe. Verificarea o folosim folosind funcția isset care returnează TRUE dacă o variabilă a fost inițializată. Butonul Adaugă este definit în codul HTML ca <input type="submit" name="butonulAdaugă" value="Adaugă"> iar variabila trimisă către scriptul PHP va fi $_POST['butonulAdaugă'] = "Adaugă". Să
```

verificăm, deci, dacă variabila a fost trimisă (dacă a fost apăsat butonul)*/

```
if(isset($_POST['butonulAdauga']))
```

/* Dacă butonul a fost apăsat, se execută acest cod. Dacă nu, scriptul ignoră codul dintre parantezele {} lui if și trece mai departe */

```
$_SESSION['titlu'][] = $_POST['titlu'];
$_SESSION['autor'][] = $_POST['autor'];
$_SESSION['nr_buc'][] = $_POST['nr_buc'];
$_SESSION['pret'][] = $_POST['pret'];
```

/* Am adăugat toate datele trimise în POST în variabile de sesiune */

/* Să afișăm acum formularul a cărui acțiune va fi același fișier:*/

```
>
<form action="test.php"
      method="post">
<table>
```

Titlu:	<input name="titlu" type="text"/>
Autor:	<input name="autor" type="text"/>
Pret:	<input name="pret" type="text"/>
Nr buc:	<input name="nr_buc" type="text"/>
	<input type="submit" value="Adaugă"/>

/* Iar în final afișăm toate cărțile din această sesiune, folosind câte un rând de tabel <tr> pentru fiecare carte: */

```
print '<table border="1"
      cellspacing="0" cellpadding="4">';
```

Titlu	Autor	Pret	Nr buc	Total
Un veac de singurătate	Gabriel Garcia Marquez	125300	2	250600

Formularul cu ajutorul căruia vom stoca date în variabile de sesiune și tabelul în care afișăm variabilele stocate deja în memorie.

Titlu	Autor	Pret	Nr buc	Total
Un veac de singurătate	Gabriel Garcia Marquez	125300	2	250600
alt titlu	alt autor	254157	3	762471

Datele trimise cu ajutorul formularului au fost adăugate în variabile de sesiune iar apoi afișate în tabel.

```
$nrCarti=count($_SESSION['titlu']);
for($i = 0; $i < $nrCarti; $i++)
{
    print '<tr>';
    print '<td>'. $_SESSION['titlu'][$i]
          .'</td>';
    print '<td>'. $_SESSION['autor'][$i]
          .'</td>';
    print '<td>'. $_SESSION['pret'][$i]
          .'</td>';
    print '<td>'. $_SESSION['nr_buc'][$i]
          .'</td>';
}
```

/* calculăm și cât este totalul pentru această carte, pentru Poezii va fi 1 buc x 100000 lei = 100000 lei în timp ce pentru Manualul de geografie va fi 30 buc x 50000 lei = 1500000 lei*/

```
$total = $_SESSION['nr_buc'][$i] *
        $_SESSION['pret'][$i];
print '<td>'. $total.'</td>';
print '</tr>';
```

/* înainte de a încheia loop-ul, vom adăuga această valoare la totalul general. Variabila \$totalGeneral va fi inițializată la prima rulare a loop-ului și va avea

valoarea totalului primei cărți. La a doua rulare a loop-ului va avea această valoare plus valoarea totalului celei de-a doua cărți și aşa mai departe până când se termină loop-ul. Atunci valoarea sa va fi suma totalurilor. */

```
$totalGeneral = $totalGeneral + $total;
}
print '</table>';
print 'Total General: '.$totalGeneral;
?>
```

Cum am putea folosi același fișier și pentru a modifica sau a șterge cărți din listă? Putem include tabelul cu lista cărților într-un formular și să folosim elemente de tip <input type="text"> pentru valorile care specifică numărul de bucați pentru fiecare carte în parte. Ne vom folosi de faptul că elementele array-ului \$_SESSION['pret'] sunt ordonate crescător, începând de la 0 pentru a specifica în elementele input și indexul numeric al array-ului care ne va fi folosit pentru a opera modificările.

```
<?
// Pornim sesiunea
session_start();
```

/* Dacă a fost apăsat butonul „Adaugă” executăm acest cod pentru a adăuga cărti în listă, iar dacă nu, acest cod este ignorat: */

```
if(isset($_POST['butonulAdauga']))
{
    $_SESSION['titlu'][] = $_POST['titlu'];
    $_SESSION['autor'][] = $_POST['autor'];
    $_SESSION['nr_buc'][] = $_POST['nr_buc'];
    $_SESSION['pret'][] = $_POST['pret'];
}
```

/* Dacă a fost apăsat butonul „Modifică” executăm acest cod pentru a modifica valori, iar dacă nu, codul următor este ignorat. Dacă a fost apăsat, înseamnă că s-au trimis valorile pentru câmpurile nr_buc. Folosim indexul numeric speci-

Titlu	Autor	Pret	Nr buc	Total
Un veac de singurătate	Gabriel Garcia Marquez	125300	2	250600
alt titlu	alt autor	254157	3	762471
Total General: 1013071				

Tabelul afișează toate variabilele de sesiune.

ficat atunci când vom fi creat câmpurile <input> ale formularului.*/

```
if(isset($_POST['butonulModifica'])) {
    for($i = 0; $i < count($_SESSION['titlu']); $i++) {
        $_SESSION['nr_buc'][$i] =
            $_POST['nr_buc'][$i];
    }
}
```

/* Am folosit for pentru a trece prin toate elementele array-ului \$_SESSION['nr_buc'] deoarece este cel mai simplu: toate valorile se trimit prin formular în diferent dacă sunt modificate sau nu. Noi le actualizăm pe toate, cu valorile modificate sau nu.*/

/* Să afișăm acum formularul a căruia acțiune va fi același fișier:*/

```
?>
<form action="test.php" method="post">


```

/* Iar în final afișăm toate cărțile din această sesiune într-un tabel, folosind câte un rând de tabel <tr> pentru fiecare carte: */

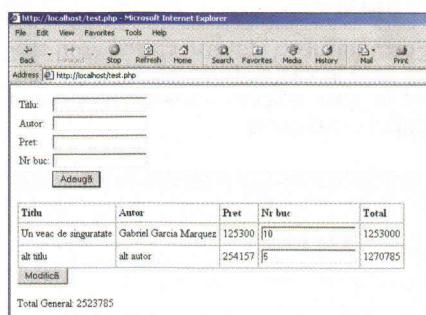
```
?>
<form action="test.php" method="post">





```

```
<td><b>Titlu</b></td>
<td><b>Autor</b></td>
<td><b>Pret</b></td>
<td><b>Nr buc</b></td>
<td><b>Total</b></td>
</tr>'>
$nrCarti=count($_SESSION['titlu']);
for($i = 0; $i < $nrCarti; $i++) {
    print '<tr>';
    print '<td>'.$_SESSION['titlu'][$i].'
<td>'.$_SESSION['autor'][$i].'
<td>'.$_SESSION['pret'][$i].'
<td>';
```



Folosind formularul putem modifica valorile stocate deja în sesiune.

```
print '<td><input type="text" name="nr_buc['.$i.'].' value="'.$_SESSION['nr_buc'][$i].'"></td>'>
```

/* calculăm și cât este totalul pentru această carte, pentru Poezii va fi 1 buc x 100000 lei = 100000 lei în timp ce pentru Manualul de geografie va fi 30 buc x 50000 lei = 1500000 lei*/

```
$total = $_SESSION['nr_buc'][$i] *
$_SESSION['pret'][$i];
print '<td>'.$total.'</td>';
print '</tr>'>
```

/* înainte de a încheia loop-ul, vom adăuga această valoare la totalul general. Variabila \$totalGeneral va fi inițializată la prima rulare a loop-ului și va avea valoarea totalului primei cărți. La a doua rulare a loop-ului va avea această valoare plus valoarea totalului celei de-a doua cărți și aşa mai departe până când se termină loop-ul. Atunci valoarea sa va fi suma totalurilor. */

```
$totalGeneral = $totalGeneral +
$total;
}
print '<table>'>
<input type="submit" name="butonulModifica" value="Modifică">
```

```
</form>
?>
print 'Total General: '.$totalGeneral;
?>
```

Să recapitulăm, folosind pseudocod, execuția scriptului nostru. El este în momentul de față împărțit în cinci părți distincte:

- pornirea/reinițializarea sesiunii;
- codul pentru adăugarea de elemente în variabilele de sesiune;
- codul pentru modificarea numărului de cărți;
- afișarea formularului pentru adăugare de cărți;
- afișarea listei de cărți și a formularului pentru modificare.

La prima rulare a scriptului, codul va fi executat astfel:

- se pornește sesiunea;
- setează codul de adăugare deoarece nu a fost apăsat butonul Adaugă (nu există variabilă \$_POST['butonulAdauga']);
- se trece și peste codul de modificare deoarece n-a fost apăsat butonul Modifică (nu există variabilă \$_POST['butonulModifica']);

- se afișează formularul;

- se afișează lista de cărți;

Dacă utilizatorul a completat formularul pentru adăugare, scriptul este rulat astfel:

- se reinițializează sesiunea;
- se execută codul de adăugare (există variabilă \$_POST['butonulAdauga']);
- se trece peste codul de modificare deoarece nu a fost apăsat butonul „Modifică” (nu a fost transmisă variabilă \$_POST['butonulModifica'] deoarece este în alt formular!);

- se afișează formularul;

- se afișează lista de cărți, inclusiv ultima care tocmai a fost introdusă. Pentru câmpul pret avem un element de tip input care specifică indexul numeric al array-ului, astfel:

- pentru prima carte:

```
<input type="text" name="nr_buc[0]" value="10">
```

- pentru a doua carte:

```
<input type="text" name="nr_buc[1]" value="5">
```

și a treia carte:

```
<input type="text" name="nr_buc[2]" value="34">
```

Indexul numeric dintre parantezele lui nr_buc[] este același cu cel din vari-

abila de sesiune `$_SESSION['nr_buc']`] și astfel va fi util pentru a actualiza valoarele stocate în sesiune.

Să presupunem că utilizatorul modifică numărul de bucăți pentru prima carte, și în loc să cumpere 10, cumpără doar 7. Atunci când apasă butonul „Modifică”, execuția scriptului va fi:

- se reinicializează sesiunea;
- se trece peste codul de adăugare (nu există variabila `$_POST['butonulAdauga']`);

- se execută codul pentru modificare. Din formular ne-au fost transmise valorile pentru trei cărți, `$_POST['nr_buc'][0]=7` (modificat în formular), `$_POST['nr_buc'][1]=5` și `$_POST['nr_buc'][2]=34`.

Trecem prin fiecare element al array-ului folosind `for` și actualizăm toate valorile, modificate sau nu (este mai ușor așa decât să verificăm care au fost modificate și să le actualizăm doar pe acelea). Atunci:

```
$_SESSION['nr_buc'][0]=$_POST['nr_buc'][0];
// 7, se modifică
$_SESSION['nr_buc'][1]=$_POST['nr_buc'][1];
// 5, nu se modifică dar este actualizat oricum
$_SESSION['nr_buc'][2]=$_POST['nr_buc'][2];
// 34, nu se modifică dar este actualizat oricum
```

- se afișează formularul de adăugare;
- se afișează lista de cărți cu modificările efectuate.

Cel mai indicat ar fi să scrieți chiar voi scriptul pentru `test.php`, să îl rulați, să vă uitați în sursa documentului HTML (View Source) din când în când (mai ales la formularul de modificare!).

Butonul pe care vor apăsa utilizatorii atunci când doresc să cumpere.

La cumpărături

Pentru coșul nostru de cumpărături vom folosi aproape același cod, cu singura diferență că formularul pentru adăugarea cărții în listă se va afla în pagina de carte. Deschideți `carte.php` și adăugați următorul formular, între informațiile despre carte și opiniiile utilizatorilor:

```
<form action=
    "cos.php?actiune=adauga"
    method="POST">
<INPUT type="hidden"
    name="id_carte"
    value="<?=$rowCarte['id_carte']?>">
<INPUT type="hidden"
    name="titlu" value=
    "<?=$rowCarte['titlu']?>">
<INPUT type="hidden"
    name="nume_autor" value=
    "<?=$rowCarte['nume_autor']?>">
<INPUT type="hidden"
    name="pret" value=
    "<?=$rowCarte['pret']?>">
<INPUT type="submit"
    value="Cumpără acum!">
</form>
```

Acest formular va arăta un singur buton pe pagină, „Cumpără acum”, dar trimite și informațiile care ne interesează, din câmpurile ascunse (hidden) prin metoda POST. Pe lângă acestea va mai trimite și o variabilă de tip GET, `$_GET['actiune'] = "adauga"` (din URL-ul `action="cos.php?actiune=adauga"`). Vom folosi această variabilă pentru a delimita codul de adăugare a informațiilor în variabile de sesiune. Similar, în formularul de modificare a cărților din listă vom avea `action="cos.php?actiune=modifica"`. Iată codul pentru `cos.php`:

```
cos.php
<?
session_start();
include("conectare.php");
include("page_top.php");
include("meniu.php");
$actiune = $_GET['actiune'];

/* Dacă este setată variabila
$_GET['actiune'] și valoarea acestaie
este "adauga", se execută acest cod: */
```

```
if(isset($_GET['actiune']) &
    $_GET['actiune'] == "adauga")
{
    $_SESSION['id_carte'][]=$_POST['id_carte'];
    $_SESSION['nr_buc'][] = 1;
    $_SESSION['pret'][]=$_POST['pret'];
}
```

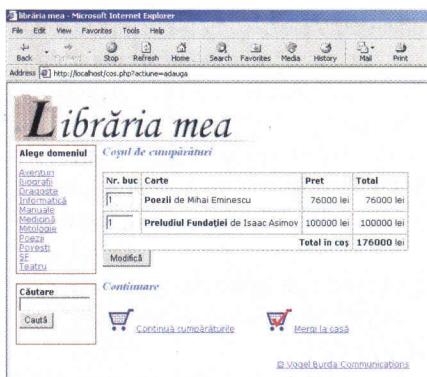
```
$_SESSION['titlu'][]=$_POST['titlu'];
$_SESSION['nume_autor'][] =
    $_POST['nume_autor'];
}
```

/* Dacă este setată variabila
`$_GET['actiune']` și valoarea acesteia
este „modifica”, se execută acest cod: */

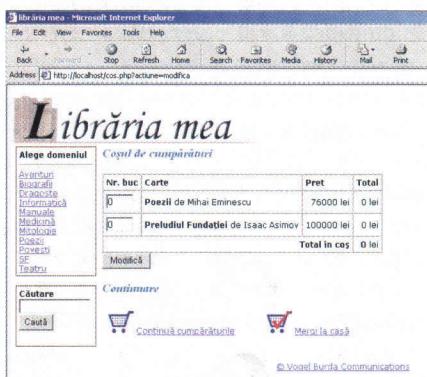
```
if(isset($_GET['actiune']) &&
    $_GET['actiune'] == "modifica")
{
    for($i=0;$i<count($_SESSION['id_carte']);
        $i++)
    {
        $_SESSION['nr_buc'][$i] =
            $_POST['noulNrBuc'][$i];
    }
}
?>
<td valign="top">
<h1>Coșul de cumpărături</h1>
<FORM action="cos.php?actiune=modifica"
    method="POST">
<table border="1" cellspacing="0"
    cellpadding="4">
<tr bgcolor="#F9F1E7">
<td><b>Nr. buc</b></td>
<td><b>Carte</b></td>
<td><b>Pret</b></td>
<td><b>Total</b></td>
</tr>
<?
for($i = 0; $i <
    count($_SESSION['id_carte']); $i++)
{
    print '<tr><td><input type="text'
        name="noulNrBuc['.$i.']" size="1"
        value="'. $_SESSION['nr_buc'][$i].'">
    </td><b>'. $_SESSION['titlu'][$i]. '</b>
    de '. $_SESSION['nume_autor'][$i]. '</td>
    <td align="right">'.
        $_SESSION['pret'][$i]. ' lei</td>
    <td align="right">'.
        ($_SESSION['pret'][$i] *
        $_SESSION['nr_buc'][$i]). '
    ' . $_SESSION['nr_buc'][$i].
    '</td>
    </tr>';
    $totalGeneral = $totalGeneral +
        ($_SESSION['pret'][$i] *
        $_SESSION['nr_buc'][$i]);
}
```

// și totalul general:

```
print '<tr><td align="right"
    colspan="3"><b>Total în cos</b></td>
    <td align="right"><b>'. $totalGeneral.
    '</b> lei</td></tr>';
?>
</table>
```



Coșul de cumpărături.



Cărțile „șterse” din coșul de cumpărături sunt încă afișate, chiar dacă totalul este 0.

```
<input type="submit"
       value="Modifică"><br><br>
Introduceți <b>0</b> pentru cărțile ce dorîți să le scoateți din coș!
<h1>Continuare</h1>
<table>
  <tr><td width="200" align="center">
    
    <a href="index.php">Continuă cumpărăturile</a></td>
    <td width="200" align="center">
      
      <a href="casa.php">Mergi la casă</a>
    </td></tr>
</table>
</td>
<?
include("page_bottom.php");
?>
```

Navigați prin paginile librăriei virtuale, adăugați cărți în coș, modificați numărul de cărți pentru a vedea cum funcționează. Mai avem o problemă de rezolvat: cum scoatem cărți din coș? Putem pune numărul de bucăți ca 0 pentru cartea pe care nu dorim să o cumpărăm însă cartea tot apare în listă. Putem condiționa afișarea cărților din loop-ul for: fiecare rând este afișat doar dacă numărul de bucăți nu este 0. În consecință vom modifica loop-ul pentru a specifica și condiția de afișare:

```
for($i = 0; $i <
    count($_SESSION['id_carte']); $i++)
{
  if($_SESSION['nr_buc'][$i] != 0)
    /* doar dacă numărul de bucăți nu e 0, afișează rândul*/
  {
    print '<tr>
      <td><input type="text"
                    name="noulNrBuc[.$i.]"
                    size="1"
                    value="'. $_SESSION['nr_buc'][$i].'">
      </td>
      <td><b>'. $_SESSION['titlu'][$i].
        '</b> de
        '. $_SESSION['nume_autor'][$i].
      </td>
      <td align="right">
        '. $_SESSION['pret'][$i]. ' lei</td>
      <td align="right">
        .' . $_SESSION['nr_buc'][$i] *
        $_SESSION['nr_buc'][$i].
        ' lei</td></tr>';
    $totalGeneral = $totalGeneral +
      ($_SESSION['pret'][$i] *
      $_SESSION['nr_buc'][$i]);
  }
}
```

Acum cărțile cu 0 ca număr de bucăți nu vor mai fi afișate.

Spunem la începutul acestui Special că vom pune pe toate paginile site-ului o casetă care să conțină valoarea însumată a cărților din coș, pentru o mai bună orientare a utilizatorului. Această casetă o vom pune în meniu din stânga. Deschideți meniu.php și adăugați următorul cod înainte de </td>:



Informațiile afișate în pagina coșului de cumpărături sunt consistente cu cele afișate în meniu

```
<br>
<div style="width:120px; background-color:#F9F1E7; padding:4px; border:solid #632415 1px">
<b>Coș</b><br>
<?
$nrCarti = 0;
$totalValoare = 0;
for($i = 0; $i <
    count($_SESSION['titlu']); $i++)
{
  $nrCarti = $nrCarti +
    $_SESSION['nr_buc'][$i];
  $totalValoare = $totalValoare +
    ($_SESSION['nr_buc'][$i] *
    $_SESSION['pret'][$i]);
}
?>
Aveți <b><?=$nrCarti?></b> cărți în coș, în valoare totală de
<b><?=$totalValoare?></b> lei.
<A href="cos.php">Click aici pentru a vedea conținutul coșului!</a>
</div>
```

Acum numărul de cărți din coș va fi disponibil pe fiecare pagină unde avem și meniu. Trebuie să adăugați session_start la începutul fiecărei pagini pe care apare meniu deoarece lucrăm cu variabile de sesiune și nu putem inițializa sesiunea în meniu deoarece până atunci se vor fi transmis date către browser. Deschideți index.php, domeniu.php și carte.php și în fiecare adăugați la început session_start:

```
<?
session_start();
include("conectare.php");
include("page_top.php");
...;
```

Numărul și valoarea cărților din coș sunt afișate tot timpul deoarece păstrăm aceste informații în variabile de sesiune.

CHIP SPECIAL – SITE DINAMIC

La casă

Mai avem de făcut un singur lucru: plăta. Pentru aceasta vom face o pagină unde utilizatorul completează un formular cu numele și adresa unde dorește să primească comanda și apoi altă pagină care verifică dacă toate câmpurile formularului au fost completate, introduce informațiile în baza de date, trimite administratorului un email și afișează utilizatorului un mesaj de mulțumire. Să facem întâi formularul:

```
casa.php
<?
session_start();
include("conectare.php");
include("page_top.php");
include("meniu.php");
?>
<td valign="top">
<h1>Casa</h1>
Acestea sunt cărțile comandate de
dvs.:





```

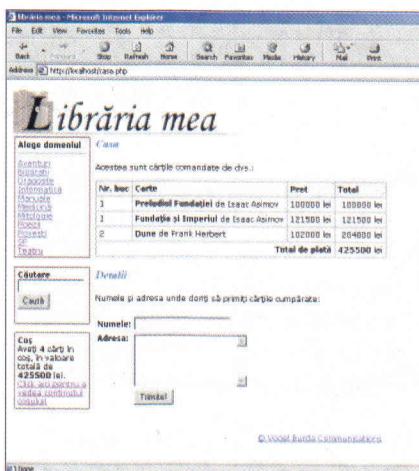
```
</tr>';
?>
</table>
<h1>Detalii</h1>
```

Numele și adresa unde doriți să primiți cărțile cumpărăte:

```
<form action="prelucrare.php"
method="POST">
<table>
<tr>
<td><b>Numele:</b></td>
<td><input type="text"
name="nume"></td>
</tr>
<tr>
<td valign="top"><b>Adresa:</b></td>
<td><textarea name="adresa"
rows="6"></textarea></td>
</tr>
<tr>
<td></td>
<td><input type="submit"
value="Trimite!"></td>
</tr>
</table>
</form>
</td>
<?
include("page_bottom.php");
?>
```

Fișierul care prelucrează aceste informații, prelucrare.php, va executa următoarele acțiuni:

- va verifica dacă numele este completat;
- va verifica dacă adresa este completată;
- va verifica numărul cărților din coș astfel încât utilizatorul să nu trimită for-



La casă: numele și adresa unde utilizatorul va primi cărțile comandate.

mularul fără să fi cumpărat vreo carte;

- se va conecta la baza de date;
- va introduce o nouă înregistrare în tabelul tranzactie cu numele, adresa și

data tranzacției;

- va lua id_tranzactie corespunzător acestei înregistrări;
- va folosi acest id_tranzactie pentru a introduce în tabelul vânzări cărțile comandate;
- vă va trimite un email de notificare;
- „curăță” sesiunea, stergând toate datele salvate în ea deoarece nu mai e nevoie de ele și ar putea induce confuzie în utilizator;
- afișează utilizatorului pagina, cu un mesaj de mulțumire.

Înainte de a trece la scrierea codului, trebuie să clarificăm un lucru: funcția mail. Pe sistemele Linux aceasta folosește sendmail sau qmail pentru a trimite mesaje, pe sistemele Windows folosește SMTP. Dacă sunteți pe o mașină Windows 2000 sau XP puteți porni serviciul SMTP pentru a trimite emailuri dar dacă aveți Windows 98, 95 sau NT va trebui să modificați setările din php.ini (din directorul Windows) și să folosiți SMTP-ul ISP-ului pentru a trimite emailuri (cel pe care îl folosiți și în clientul de mail). Singurul inconvenient pentru trimitera de emailuri prin SMTP-ul ISP-ului este că trebuie să fiți conectați la Internet, altfel veți primi un mesaj de eroare. Iată cum arată setarea SMTP în php.ini la mine:

```
[mail function]
; For Win32 only.
SMTP = mail.isp.ro
```

Dacă sunteți pe o mașină Windows 2000 sau XP puteți seta SMTP = localhost, dar serviciul de SMTP trebuie să fie pornit! Faceți modificarea în php.ini apoi faceți un fișier de test, numit mail.php:

```
mail.php
<?
mail("oana@chip.ro", "test mesaj
smtp", "testing");
print "ok";
?>
```

Înlocuiți adresa mea de email cu a dvs. (dvs. trebuie să primiți emailul, nu eu) și rulați scriptul în browser. Dacă nu sunteți conectați la Internet, adresa serverului SMTP este greșită sau serviciul SMTP nu este pornit la adresa respectivă, veți primi un mesaj de eroare, altfel veți primi mesajul la adresa dvs. de email.

Desigur, nu este foarte convenabil să trebuiască să fiți conectați la Internet de câte ori doriți să trimiteți un email prin

intermediul PHP.

Din fericire, avem o soluție simplă pentru această problemă: putem pune operatorul @ în fața funcției mail și astfel împăcăm și capra și varza: dacă poate trimite mail îl trimite, dacă nu, nu returnează nici un mesaj de eroare (dar nici nu va trimite emailul).

mail.php

```
<?
@mail("oana@chip.ro","test mesaj
smtp","testing");
print "ok";
?>
```

Astfel putem lucra și testa în continuare aplicația offline, iar dacă la un moment dat dorim să o transferăm pe un server care poate trimite email, nu va mai trebui să facem nici un fel de modificări.

Cu acestea lămurite, să scriem codul pentru fișierul prelucrare.php

prelucrare.php

```
<?
// Verificăm dacă numele este completat, iar dacă nu e, oprim execuția scriptului
if($_POST['nume'] == "")
{
    print 'Trebuie să completați numele!
    <a href="cos.php">înapoi</a>';
    exit;
}
```

/* Verificăm dacă adresa este completată iar dacă nu e, oprim execuția scriptului */

```
if($_POST['adresa'] == "")
{
    print 'Trebuie să completați adresa!
    <a href="cos.php">înapoi</a>';
    exit;
}
```

/* Reinițializăm sesiunea deoarece dorim să verificăm numărul de cărți comandate */

```
session_start();
```

/* numărul de cărți comandate îl aflăm rapid, folosind array_sum(array_sum(\$array) returnează suma valorilor dintr-un array dacă acestea sunt numerice. Astfel, dacă \$a = array('1','1','2'), array_sum(\$a) = 4. Nu confundați array_sum cu count, count(\$a) = 3 elemente în timp ce array_sum(\$a) = 4 (suma elementelor). */

```
$nrCarti=array_sum($_SESSION['nr_buc']);
if($nrCarti == 0)
{
    print 'Trebuie să cumpărați cel
    puțin o carte! <a
    href="cos.php">înapoi</a>';
    exit;
}
```

/* În acest moment toate datele sunt verificate, putem să ne conectăm la baza de date pentru a le introduce: */

```
include("conectare.php");
```

/* Introducem întâi datele în tabelul tranzactii. Deoarece câmpul data din tabel este de tip TIMESTAMP, îl putem omite (se va seta singur, cu data curentă) */

```
$sqlTranzactie = "insert into
    tranzactii(nume_cumparator,
    adresa_cumparator)
values ('".$_POST['nume'].",
    '".$_POST['adresa']."'')";
$resursaTranzactie =
    mysql_query($sqlTranzactie);
```

/* Obținem id-ul acestei înregistrări folosind mysql_insert_id(): */

```
$id_tranzactie = mysql_insert_id();
```

/* iar acum luăm fiecare carte din sesiune și o introducем în tabelul vânzări. Introducem în tabel doar cărțile ale căror număr de bucăți este mai mare ca 0 (în condiția if, din cadrul structurii for): */

```
for($i=0;$i<count($_SESSION['id_carte']);
    $i++)
{
    if($_SESSION['nr_buc'][$i] > 0)
    {
        /* Creăm interogarea */
        $sqlVanzare = "INSERT INTO vanzari
            values ('".$id_tranzactie."','".$'.
            $_SESSION['id_carte'][$i]."',".
            $_SESSION['nr_buc'][$i]."')";

        /* și o rulăm */
        mysql_query($sqlVanzare);
    }
}
```

/* Urmează să trimitem un email de notificare folosind funcția mail. mail folosește în principal trei argumente: mail(adresa destinatarului, subiectul mesajului, textul mesajului) dar mai poate prelua încă unul, pentru headere adiționale. */

```
$emailDestinatar = "oana@chip.ro";
```

/* schimbăți adresa cu cea la care doriți să primiți mesajele */

```
$subiect = "O nouă comandă!";
```

/* Pentru a compune mesajul ne vom folosi de operatorul .= de concatenare a stringurilor. */

```
$mesaj = "O nouă comandă de la
    <b>". $_POST['nume'] . "</b><br>";
```

```
$mesaj .= "Adresa:
```

```
    ". $_POST['adresa'] . "<br>";
```

```
$mesaj .= "Cartile
    comandate:<br><br>";
```

```
$mesaj .= "<table border='1'
    cellspacing='0' cellpadding='4'>;
for($i=0;$i<count($_SESSION['id_carte']);
    $i++)
{
    if($_SESSION['nr_buc'][$i] > 0)
```

```
    {
        $mesaj .= "<tr><td>".
            $_SESSION['titlu'][$i]. " de
            ".$_SESSION['nume_autor'][$i].
            "</td><td>".$_SESSION['nr_buc'][$i].
            " buc</td></tr>";
```

```
$totalGeneral = $totalGeneral +
    ($_SESSION['nr_buc'][$i] *
    $_SESSION['pret'][$i]);
}
```

```
}
```

```
$mesaj .= "</table>";
```

```
$mesaj .= "Total:
```

```
    <b>". $totalGeneral . "</b>";
```

/* Punem headere adiționale pentru a trimite mesajul în format HTML și encodingul potrivit pentru caracterele românești: */

```
$headers = "MIME-Version:
    1.0\r\nContent-type: text/html;
    charset=iso-8859-2\r\n";
```

/* Acum putem trimite emailul: */

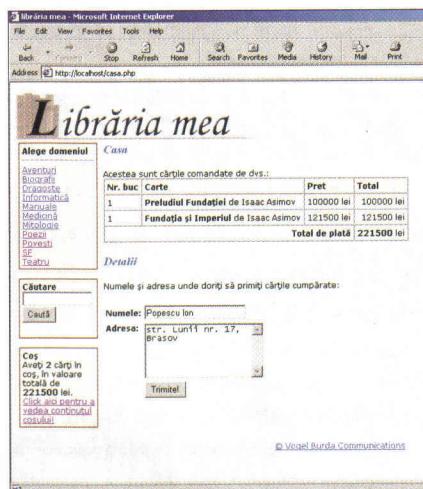
```
mail($emailDestinatar, $subiect,
    $mesaj, $headers);
```

/* Curățăm sesiunea deoarece nu mai avem nevoie de datele din ea.*/

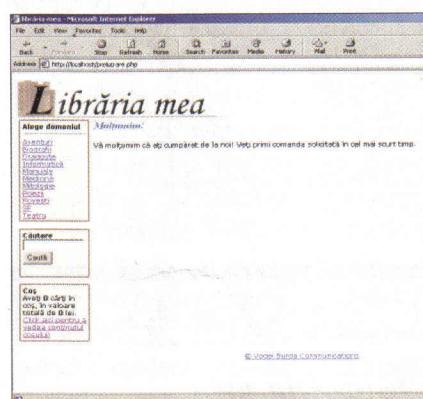
```
session_unset();
```

/* eliminăm toate variabilele asociate acestei sesiuni */

```
session_destroy();
```



Utilizatorul completează formularul cu informațiile personale.

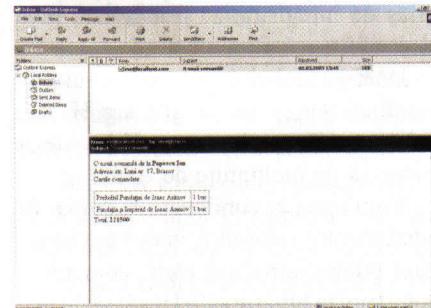


Mesajul de confirmare și mulțumire

```
/* ștergem sesiunea */
/* În final afișăm utilizatorului pagina cu mesajul de mulțumire: */


```

```
include("page_top.php");
include("meniu.php");
?>
<td valign="top">
<h1>Mulțumim!</h1>
Vă mulțumim că ați cumpărat de la noi! Veți primi comanda solicitată în cel mai scurt timp.
</td>
?>
include("page_bottom.php");
?>
```



La fiecare nouă comandă, putem primi email de notificare.

Căutare

Mai avem o singură pagină de făcut: cea de căutare. Vom lăsa posibilitatea utilizatorului să caute în baza noastră de date și să îi afișeze rezultatele pe o pagină. Să deschidem întâi meniu.php unde se află formularul pentru căutare. Observați că are metoda GET (<form action="cautare.php" method="GET">). De ce? Pentru că atunci vom putea adăuga în lista Favorites a browserului rezultatele unei căutări. Cu metoda POST, adresa paginii de căutare după apăsarea butonului "Caută" ar fi http://localhost/cautare.php dar cu GET ea va fi http://localhost/cautare.php?cuvant=poezii. Acest URL îl putem apoi trimite altciva pentru a vedea direct rezultatele căutării sau îl putem adăuga la lista de Favorites pentru mai târziu.

Înainte de a scrie codul, să ne aducem aminte care era comanda SQL pentru selectarea parțială a unui cuvânt. Foloseam wildcard-ul %, astfel: SELECT * FROM table WHERE coloana LIKE "%cuvant%". Aceeași comandă SQL o vom folosi și în continuare pentru a interoga tabelele carti și autori după cuvântul sau cuvintele scrise de utilizator:

```
<?
session_start();
include("conectare.php");
include("page_top.php");
include("meniu.php");
$cuvant = $_GET['cuvant'];
?>
<td valign="top">
<h1>Rezultatele căutării</h1>
<p>Textul căutat: <b><?=$cuvant?></b></p>
<b>Autori</b>
<blockquote>
```

/* Interogăm întâi tabelul autorilor:*/

```
<?
$sql = "SELECT id_autor, nume_autor
FROM autori WHERE nume_autor LIKE
'%".$cuvant."%'";
$resursa = mysql_query($sql);
```

/* Dacă interogarea nu a returnat nici un rezultat (0 rânduri) scriem "Nici un rezultat" */

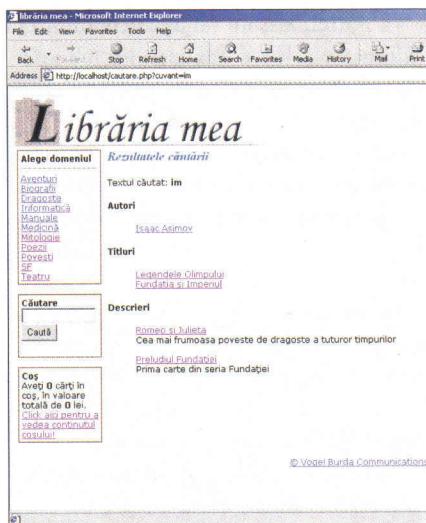
```
if(mysql_num_rows($resursa) == 0)
{
    print "<i>Nici un rezultat</i>";
}
```

/* Altfel, afișăm rândurile rezultate*/

```
while($row=mysql_fetch_array($resursa))
{
    print '<a href="autor.php?
id_autor='.$row['id_autor'].'">'.
$row['nume_autor'].'</a><br>';
}
?>
</blockquote>
<b>Titluri</b>
<blockquote>
<?
$sql="SELECT id_carte, titlu FROM carti
WHERE titlu LIKE '%".$cuvant."%'";
$resursa = mysql_query($sql);
if(mysql_num_rows($resursa) == 0)
{
    print "<i>Nici un rezultat</i>";
}
while($row=mysql_fetch_array($resursa))
{
    print '<a href="carte.php?id_carte='.
$row['id_carte'].'">'.
$row['titlu'].'</a><br>';
}
?>
</blockquote>
<b>Descrieri</b>
<blockquote>
<?
$sql = "SELECT id_carte, titlu,
descriere FROM carti WHERE
descriere LIKE '%".$cuvant."%'";
$resursa = mysql_query($sql);
if(mysql_num_rows($resursa) == 0)
{
    print "<i>Nici un rezultat</i>";
}
while($row=mysql_fetch_array($resursa))
{
    print '<a
href="carte.php?id_carte='.
$row['id_carte'].'">'.
$row['titlu'].'</a><br>'.
$row['descriere'].'<br><br>';
}
?>
</blockquote>
</td>
?>
include("page_bottom.php");
?>
```

Observați că am făcut referire la un fișier inexistent, autor.php. Aceasta va fi tema voastră. Un click pe numele lui Mihai Eminescu în pagina de căutare ne va duce la pagina http://localhost/autor.php?id_autor=1.

Folosiți \$_GET['id_autor'] pentru a interoga tabelul cărți și a obține o listă cu



Rezultatele căutării pe site

cărțile scrise de autorul respectiv. Luați pagina domeniu.php ca exemplu.

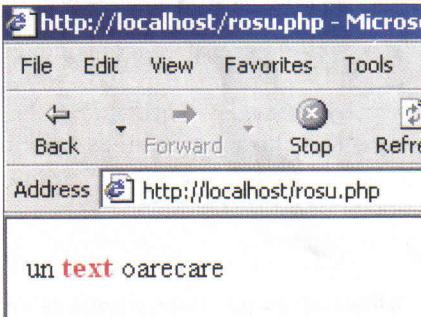
Să revenim la căutarea noastră. Cum am putea afișa rezultatele astfel încât cuvântul sau textul căutat să fie evidențiat, la fel ca în Google? Putem folosi funcția str_replace pentru a înlocui o parte dintr-un string. Funcția se folosește în felul următor:

```
$stringModificat = str_replace
("textul vechi din string ce urmează a fi modificat", "textul nou,
modificat", $stringVechi);
```

Iată într-un mic exemplu, cum se folosește această funcție:

rosu.php

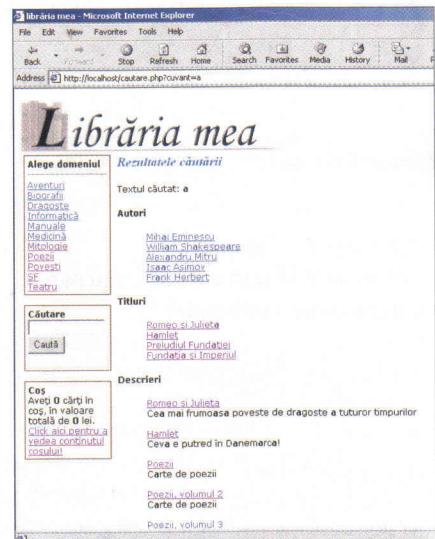
```
<?
$text = "un text oarecare";
$text = str_replace("text","<b><font
color='red'>text</font></b>", $text);
print $text;
?>
```



Exemplu str_replace.

Aceeași funcție o vom folosi și înainte de a afișa rezultatele căutării, pentru a evidenția (bold) textul căutat. Iată fișierul cautare.php modificat astfel încât să fie evidențiat textul căutat:

```
<?
session_start();
include("conectare.php");
include("page_top.php");
include("meniu.php");
$cuvant = $_GET['cuvant'];
?>
<td valign="top">
<h1>Rezultatele căutării</h1>
<p>Textul căutat: <b><?= $cuvant ?>
</b></p>
<b>Autori</b>
<blockquote>
<?
$sql = "SELECT id_autor, nume_autor
FROM autori WHERE nume_autor LIKE
'%'. $cuvant . '%'";
$resursa = mysql_query($sql);
if(mysql_num_rows($resursa) == 0)
{
    print "<i>Nici un rezultat</i>";
}
while($row =
mysql_fetch_array($resursa))
{
    $nume_autor = str_replace
    ($cuvant, "<b>$cuvant</b>",
    $row['nume_autor']);
    print '<a href="autor.php?
id_autor='.$row['id_autor'].'">
    '.$nume_autor.'</a><br>';
}
?>
</blockquote>
<b>Titluri</b>
<blockquote>
<?
$cuvant = str_replace
($cuvant, "<b>$cuvant</b>",
$row['titlu']);
print '<a href="carte.php?id_carte='
.$row['id_carte']. '">' .
.$row['titlu']. '</a><br>' .
.$resursa['descriere']. '<br><br>';
}
?>
</blockquote>
<b>Descrieri</b>
<blockquote>
<?
$sql = "SELECT id_carte, titlu FROM carti
WHERE titlu LIKE '%'. $cuvant . '%'";
$resursa = mysql_query($sql);
if(mysql_num_rows($resursa) == 0)
{
    print "<i>Nici un rezultat</i>";
}
while($row =
mysql_fetch_array($resursa))
{
    $titlu = str_replace
    ($cuvant, "<b>$cuvant</b>",
    $row['titlu']);
    print '<a href="carte.php?
id_carte='.$row['id_carte'].'">
    '.$titlu.'</a><br>';
}
?>
</blockquote>
<b>Descrieri</b>
<blockquote>
<?
$sql = "SELECT id_carte, titlu, descriere
FROM carti WHERE descriere
LIKE '%'. $cuvant . '%'";
$resursa = mysql_query($sql);
if(mysql_num_rows($resursa) == 0)
{
    print "<i>Nici un rezultat</i>";
}
while($row =
mysql_fetch_array($resursa))
{
    $descriere = str_replace
    ($cuvant, "<b>$cuvant</b>",
    $row['descriere']);
    print '<a href="carte.php?id_carte='
.$row['id_carte']. '">' .
.$row['titlu']. '</a><br>' .
.$descriere. '<br><br>';
}
?>
</blockquote>
</td>
<?
include("page_bottom.php");
?>
```



Textul căutat este evidențiat în rezultatele căutării.

Cu acestea am terminat partea „vizibilă” a site-ului nostru. După cum v-am promis la început, am făcut doar câteva fișiere care să poată afișa un număr uriaș de înregistrări... și asta fără prea mult efort.

Doar gândul că nu va mai trebui vreodată să scrieți un singur tag HTML pentru a adăuga elemente noi pe site ar trebui să fie mulțumire de ajuns!

Vom trece în continuare la partea de administrare a site-ului, unde vom vedea cum putem introduce cărți, domenii și autori noi în librărie astfel încât să nu mai avem vreodată de-a face cu chinuitoarea linie de comandă.

Administrare site

Stăpânul datelor

În acest capitol ne vom ocupa de secțiunea de administrare, începând de la securizarea ei, până la introducerea datelor, upload-ul de fișiere și manipularea de imagini.

Am pus utilizatorul pe primul plan și am făcut secțiunea „vizibilă” a site-ului nostru astfel încât informațiile să fie întotdeauna „la zi”. Este momentul să ne gândim și la noi și la timpul pe care l-am putea câștiga dacă am avea o interfață simplă și ușor de utilizat pentru manipularea diferitelor aspecte ale site-ului.

Așa cum am avut doar câteva pagini pentru a afișa mai multe cărți utilizatorilor, la fel putem avea doar câteva pagini pentru a introduce informații noi în baza de date. Vom avea mai multe pagini, câte una pentru fiecare acțiune ce dorim să o întreprindem: adăugare, modificare, ștergere, moderare comentarii utilizatori și urmărire comenzi. Vom învăța și cum să oferim acces în această zonă doar administratorului sau altor utilizatori autorizați.

Creați un director nou în document root și denumiți-l administrare. Aici vom păstra toate fișierele secțiunii de administrare, ele fiind accesibile la adresa <http://localhost/administrare/>

Autentificare

Primul lucru pe care trebuie să îl facem este să asigurăm această zonă, să nu permitem accesul decât administratorului, cu o anumită combinație de nume și parolă. Pentru aceasta vom scrie trei fișiere:

- index.php unde vom afișa formularul de login

- login.php unde verificăm informațiile transmise din formular, le coroborăm cu cele existente în baza de date și dacă sunt corecte, acordăm accesul mai departe

- autorizare.php este pagina pe căreia vom include apoi la începutul fiecărui fișier din secțiunea de administrare. Ea va verifica la fiecare accesare dacă utilizatorul este înregistrat și are acces pe pagina respectivă și permite rularea pagin-

nii doar dacă utilizatorul este înregistrat. Fără această verificare, un utilizator ar putea accesa <http://localhost/pagina.php>, fără să treacă prin formularul de înregistrare.

Să trecem la treabă. După cum spuseam, index.php conține un simplu formular:

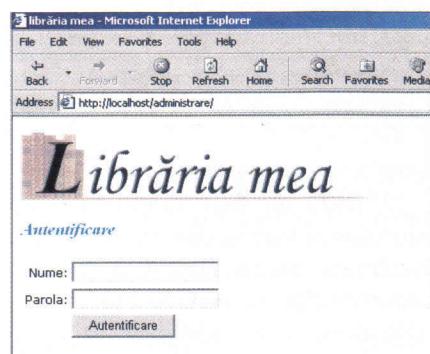
[administrare/index.php](#)

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<title>libraria mea</title>
<style type="text/css">
body, p, td {font-family: Verdana, Arial, sans-serif; font-size: 12px;}
h1 {font-family: Times New Roman, Times, serif; font-size: 18px; font-weight: bold; color: #336699; font-style: italic;}
.title {font-family: Verdana, Arial, sans-serif; font-size: 14px; font-weight: bold; color: #0066CC; }
</style>
</head>
<body bgcolor="#ffffff">

<h1>Autentificare</h1>
<form action="login.php" method="POST">
<table>
<tr>
<td align="right">Nume:</td>
<td><input type="text" name="nume"></td>
</tr>
<tr>
<td align="right">Parola:</td>
<td><input type="password" name="parola"></td>
</tr>
</table>

```

```
<td></td>
<td><input type="submit" value="Autentificare"></td>
</tr>
</table>
</form>
</body>
</html>
```



Accesul în secțiunea de administrare se face pe bază de nume și parolă.

Pentru ca în următorul fișier, login.php, să poată confrunta datele din formular cu cele din baza de date, trebuie întâi să avem informațiile stocate în baza de date. Va trebui să facem pentru aceasta un nou tabel, admin, în baza de date librarie:

```
CREATE TABLE admin (
admin_nume text NOT NULL,
admin_parola text NOT NULL
);
```

În acest tabel vom salva numele și parola pentru accesul la secțiunea de administrare. Dar, cum salvarea parolei „în clar” este în general o idee proastă, o vom introduce în tabel criptată:

```
INSERT INTO admin VALUES
("administrator", md5("parola"));
```

Iată cum arată conținutul tabelului:

SELECT * FROM admin;	
+	-----+-----+
admin_nume admin_parola	-----+-----+
+	-----+-----+
administrator 8287458823facb8ff918dbfabcd22ccb	-----+-----+
+	-----+-----+

Criptarea folosind md5 nu este reversibilă (și astfel nici dvs., nici altcineva nu o va putea afla chiar dacă are acces la baza de date). Să scriem scriptul care

să confrunte informațiile din formular cu cele din baza de date:

```
administrare/login.php
<?
/* Verificăm întâi dacă au fost completeate amândouă câmpurile */
if($_POST['nume'] == "" || $_POST['parola'] == "")
{
    print 'Trebuie să completezi amândouă câmpurile!<br>
<a href="index.php">înapoi</a>';
    exit;
}

/* Dacă au fost completate, ne conectăm la baza de date pentru a putea confrunta datele. Folosim fișierul conectare.php din document root:*/
include("../conectare.php");
/* Criptăm parola trimisă prin formular tot cu md5, pentru a o avea în formatul în care ea există în baza de date:*/
$parolaEncriptata=md5($_POST['parola']);
/* și scriem interogarea de verificare:*/
$sql = "SELECT * FROM admin WHERE
        admin_nume='".$_POST['nume']."' AND
        admin_parola='".$parolaEncriptata."'";
$resursa = mysql_query($sql);

/* Această interogare, dacă este executată cu succes (numele și parola din formular sunt valide și corespund celor din baza de date) ar trebui să returneze un singur rând din tabelul admin, care corespunde numelui și parolei. Pentru mulți programatori, verificarea returnării unui rezultat este de ajuns pentru a acorda accesul mai departe. Noi vom extinde protecția și vom acorda accesul doar dacă rezultatul returnat are un singur rând (mai precis, vom afișa un mesaj de eroare și vom opri execuția scriptului dacă numărul de rânduri nu e 1):*/
if(mysql_num_rows($resursa) != 1)
{
    print 'Nume sau parolă greșite!<br>
    <a href="index.php">înapoi</a>';
    exit;
}
```

/* Până acum s-au făcut toate verificările, numele și parola sunt corecte, să trecem la autentificarea propriu-zisă. Vom folosi variabile de sesiune pentru a păstra în memorie câteva informații despre autentificare, pentru a le reverifica mai târziu, atunci când accesăm alte pagini din cadrul secțiunii de administrare. Pornim

întâi sesiunea după care trecem la salvarea informațiilor în ea: */

```
session_start();
$_SESSION['nume_admin'] =
$_POST['nume'];
$_SESSION['parola_encriptata'] =
$parolaEncriptata;

/* Pe lângă acestea, pentru o și mai mare siguranță, vom salva id-ul sesiunii în altă variabilă. Toate sesiunile au un id unic, un string care seamănă cu rezultatul unei criptări md5.*/
$_SESSION['key_admin'] = session_id();

/* Cu autentificarea făcută, spunem scriptului să încarce prima pagină din secțiunea de administrare:*/
header("location: admin.php");
?>
```

Creați și o pagină admin.php în care scrieți textul „secțiunea de administrare”, după care accesați adresa <http://localhost/administrare/> și autentificați-vă cu numele „administrator” și parola „parola..”. Autentificarea va fi făcută și veți fi redirecționați către pagina admin.php. Să facem un alt test: închideți toate instanțele browserului astfel încât să expire sesiunea, porniți din nou un browser și accesați direct adresa <http://localhost/administrare/admin.php>. Veți avea acces, chiar dacă nu v-ați autentificat folosind formularul! Ei bine, putem împiedica accesul neautorizat dacă scriem un mic script de verificare a datelor sesiunii înainte de a încărca orice pagină din secțiunea de administrare. Iată codul:

```
autorizare.php
<?
/* Pornim sesiunea, deoarece avem nevoie de datele din ea:*/
session_start();
/* și verificăm datele, începând cu $_SESSION['key_admin'] definit în login.php care trebuie să fie același cu session_id:*/
if($_SESSION['key_admin']!=session_id())
{
    print 'Acces neautorizat!';
    exit;
}
```

/* Ne conectăm la baza de date deoarece vom reverifica datele din sesiune cu cele din baza de date: */

```
include("../conectare.php");
/* Verificăm dacă numele și parola
```

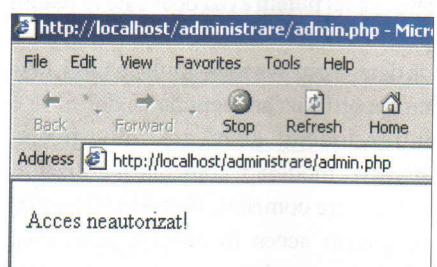
salvate în variabile de sesiune sunt aceleași cu cele din baza de date*/

```
$sql = "SELECT * FROM admin WHERE
        admin_nume='".$_SESSION['nume_admin']."' AND
        admin_parola=
        '".$_SESSION['parola_encriptata']."'";

$resursa = mysql_query($sql);
```

/* Această interogare, dacă este executată cu succes (numele și parola din variabilele de sesiune sunt valide și corespund celor din baza de date) ar trebui să returneze un singur rând din tabelul admin. Dacă nu returnează exact un rând vom afișa un mesaj de eroare și vom opri execuția scriptului:*/

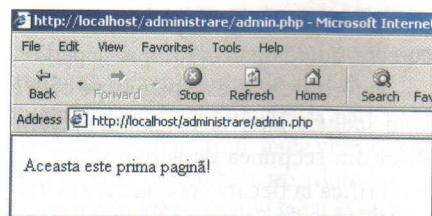
```
if(mysql_num_rows($resursa) != 1)
{
    print 'Acces neautorizat!';
    exit;
}
?>
```



Dacă numele și parola nu sunt corecte, nu se acordă acces mai departe.

Vom folosi autorizare.php la începutul fiecărei pagini și împușcăm astfel trei iepuri dintr-un foc: în acest script inițializăm sesiunea, ne conectăm la baza de date și verificăm autentificarea. Ca urmare, dacă includem acest script în orice fișier din secțiunea de administrare vom fi și gata conectați la baza de date, și cu sesiunea pornită și cu autorizarea efectuată. Să rescriem fișierul admin.php:

```
<?
include("autorizare.php");
print "Aceasta este prima pagină!";
?>
```



Dacă numele și parola sunt valide, accesul este acordat.

Închideți toate instanțele browserului pentru a șterge sesiunea (dacă v-ați logat) și apoi accesați adresa <http://localhost/administrare/admin.php> (direct, fără să treceți prin formularul de autentificare). Vom primi mesajul „Acces neautorizat!” și restul textului nu va mai fi afișat! Accesați acum <http://localhost/administrare/> și completați formularul de autentificare. Dacă le-ați completat corect, veți ajunge pe pagina admin.php și veți vedea textul „Aceasta este prima pagină!”. Paginile de pe secțiunea de administrare sunt de acum asigurate!

Pentru că zona de administrare are mai multe secțiuni, pentru adăugare, ștergere, etc., va trebui să avem link-uri către toate acele secțiuni, pe fiecare pagină. Decât să le scriem de fiecare dată când avem nevoie de ele, mai bine le scriem într-un fișier, admin_top.php pe care să îl accesăm atunci când avem nevoie și în care scriem și instrucțiunile <html>, <body>, etc. Iată conținutul fișierului:

admin_top.php

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<title>librăria mea</title>
<style type="text/css">
body, p, td {font-family: Verdana, Arial, sans-serif; font-size: 12px;}
h1 {font-family: Times New Roman, Times, serif; font-size: 18px; font-weight: bold; color: #336699; font-style: italic;}
.titlu {font-family: Verdana, Arial, sans-serif; font-size: 14px; font-weight: bold; color: #0066CC;}
</style>
</head>
<body bgcolor="#ffffff">

<TABLE border="1" cellpadding="4" cellspacing="0" style="width: 100%; border-collapse: collapse;>
<tr>
<td><A href="adaugare.php">Adaugă</a></td>
<td><A href="modificare_stergere.php">Modifică sau șterge</a></td>
<td><A href="opinii.php">Opinii</a></td>

```

```
vizitatori</a></td>
<td><A href="comenzi.php">Comenzi</a></td>
</tr>
</table>
<br>
```

Să modificăm acum și admin.php și să includem acest fișier în el:

admin.php

```
<?
include("autorizare.php");
include("admin_top.php");
print "Aceasta este prima pagină!";
?>
```



Prima pagină a secțiunii de administrare.

Uneori, includerea unui fișier care să se ocupe cu autorizarea nu este o soluție viabilă. De exemplu, putem avea următorul exemplu în care o pagină a secțiunii de administrare (test.php) include alte două pagini (meniu.php și main.php). Atunci ar trebui să includem fișierul autorizare.php în fiecare din ele, dar atunci am primi un mesaj de eroare (se reinicializează sesiunea de 3 ori și în plus, în meniu.php și main.php se reinicializează după ce au fost trimise date către browser).

Atunci putem crea o funcție declarată de noi, care să facă autorizarea.

Această funcție o putem apoi apela oricând avem nevoie de ea (cu condiția să fie definită întâi!).

Iată cum ar arăta structura site-ului în acest caz:

test.php (include autorizare.php care pornește sesiunea, se conectează la baza de date, verifică autorizarea)

meniu.php (include autorizare.php care pornește sesiunea, se conectează la baza de date, verifică autorizarea)

main.php (include autorizare.php care pornește sesiunea, se conectează la baza de date, verifică autorizarea)

Observăm că în cele două fișiere care sunt parte a lui test.php nu facem decât să verificăm autorizarea, fără să includem nici un alt fișier. Cum funcția de autorizare este definită în fișierul „mamă”, dacă meniu.php sau main.php sunt apeleate din cadrul lui, vor face verificarea folosind funcția deja definită. Iată structura celor patru fișiere:

test.php (include f_autorizare.php care pornește sesiunea, se conectează la baza de date, definește funcția de autorizare și folosește această funcție pentru a face autorizarea)

meniu.php (se apeleză funcția de autorizare și se acordă sau nu accesul)

main.php (se apeleză funcția de autorizare și se acordă sau nu accesul)

f_autorizare.php

```
<?
session_start();
include("../conectare.php");
function autorizat()
{
    $sql = "SELECT * FROM admin WHERE
    admin_nume='".$SESSION['nume_admin']."' AND
    admin_parola='".$SESSION['parola_encriptata']."' ";
    $resursa = mysql_query($sql);
    if($SESSION['key_admin'] != session_id() ||
    mysql_num_rows($resursa) != 1)
    {
        return false;
    }
    else
    {
        return true;
    }
?>
```

test.php

```
<?
/* include fișierul f_autorizare.php care pornește sesiunea, se conectează la baza de date și declară funcția de autorizare*/
include("f_autorizare.php");
```

```
/* și verifică autorizarea folosind funcția definită în fișierul pe care tocmai l-am inclus, f_autorizare.php*/
if(!autorizat())
{
```

```
    print 'Acces neautorizat!';
    exit;
```

```

)
include("meniu.php");
include("main.php");
?>

```

meniu.php

```

/* verifică doar autorizarea*/
if(!autorizat())
{
    print 'Acces neautorizat!';
    exit;
}
/* codul care re rulează dacă este
autorizat */

```

test.php (definește funcția de autorizare, folosește această funcție pentru a face autorizarea)

meniu.php, inclus în test.php (definește funcția de autorizare, folosește această funcție pentru a face autorizarea=> eroare! funcția a fost definită de două ori (o dată în fișierul „mama”, o dată aici).

main.php

```

/* verifică doar autorizarea*/
if(!autorizat())
{
    print 'Acces neautorizat!';
    exit;
}
/* codul care se rulează dacă este
autorizat */

```

Astfel vedem cum, folosind o funcție definită de noi, nu se va putea accesa nici unul din fișierele incluse fără autorizare! De cele mai multe ori, o funcție este mult mai utilă decât un fișier inclus însă alegerea este a dvs. Fiți atenți doar să nu definiți funcția de două ori în cadrul aceluiasi script, altfel veți primi mesajul de eroare **Fatal error:** Cannot redeclare numele_funcției().

Dacă funcția a fost deja definită undeva în execuția scriptului, trebuie doar să o apelați.

Adăugare

În baza de date putem adăuga trei lucruri: domenii, autori sau cărți noi. Pentru acestea vom folosi o pagină în care vom afișa trei formulare, câte unul pentru fiecare și altă pagină care va prelucra informațiile din aceste formulare, în funcție de care a fost trimis către

server. În pagina adaugare.php vom avea:

- formularul pentru introducere domeniu nou;
- formularul pentru introducere autor nou;
- formularul pentru introducere carte nouă;

Pagina prelucrare_adaugare.php va avea următoarea structură:

- include pagina admin_top.php (se pornește sesiunea, se conectează la baza de date și încarcă începutul paginii);
- dacă a fost trimis primul formular, îl prelucrează și introduce domeniul nou în tabelul domenii și apoi afișează un mesaj de confirmare;
- dacă a fost trimis al doilea formular, îl prelucrează și introduce noul autor în tabelul autori și afișează un mesaj de confirmare;
- dacă a fost trimis al treilea formular, îl prelucrează și introduce noua carte în tabelul carti și apoi afișează un mesaj de confirmare.

Înainte de a crea cele două fișiere, vom mai face un singur lucru. Ne aducem aminte că tabelul carti are un câmp de tip DATE numit data în care se stochează data adăugării cărții în baza de date. Această dată ne folosește pentru a afișa pe prima pagină cele mai noi trei cărți. Știm din experiență că este mult mai simplu să lucrăm cu câmpuri de tip TIMESTAMP (se setează automat cu data curentă), așa că ne vom conecta la baza de date și vom modifica tipul câmpului:

```

ALTER TABLE carti CHANGE data data
TIMESTAMP(10) NOT NULL;

```

Toate valorile înregistrate anterior vor fi convertite, iar de acum încolo câmpul data se va seta automat la fiecare INSERT. Altfel ar fi trebuit să îl specificăm noi.

Iată cele două fișiere:

adaugare.php

```

<?
include("autorizare.php");
include("admin_top.php");
?>

<h1>Adăugare</h1>
<b>Adăugă domeniu</b>
<form action="prelucrare_adaugare.php"
method="POST">

```

```

Domeniu nou: <INPUT type="text"
name="domeniu_nou">
<INPUT type="submit"
name="adauga_domeniu"
value="Adăugă">
</form>

```

```

<b>Adăugă autor</b>
<form
action="prelucrare_adaugare.php"
method="POST">

```

```

Autor nou: <INPUT type="text"
name="autor_nou">
<INPUT type="submit"
name="adauga_autor"
value="Adăugă">
</form>

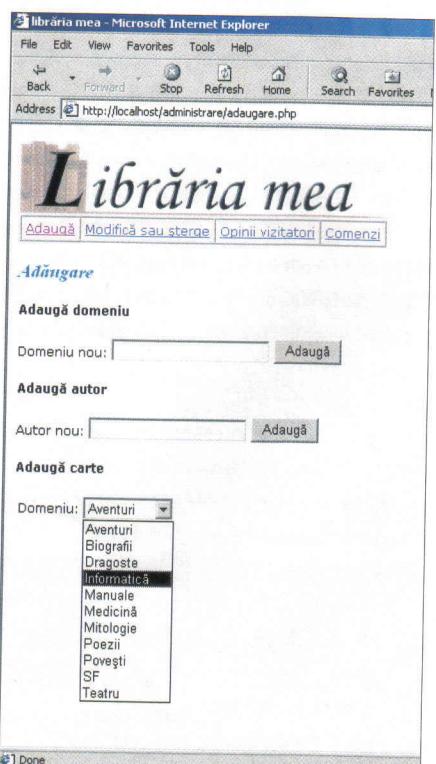
```

```

<b>Adăugă carte</b>
<form
action="prelucrare_adaugare.php"
method="POST">
<table>
<tr>
<td>Domeniu:</td>
<td>
<select name="id_domeniu">

```

/* Luăm numele de domenii din baza de date și le afișăm utilizatorului într-o listă drop-down. Astfel putem obține un id_domeniu corespunzător domeniului selectat pe care să îl introducem în tabelul carti */



Alegerea unui domeniu dintr-o listă generată automat.

```
<?
$sql = "SELECT * FROM domenii
        ORDER BY nume_domeniu ASC";
$resursa = mysql_query($sql);
while($row=mysql_fetch_array($resursa))
{
    print '<option value="'.
    "'.$row['id_domeniu'].'">'.
    $row['nume_domeniu'].'</option>';
}
?>
</select>
</td>
</tr>
<td>Autor:</td>
<td>
<select name="id_autor">
```



Librăria mea

[Adaugă](#) | [Modifică sau stergă](#) | [Opinii vizitatori](#) | [Comenzi](#)

Adăugare

Adaugă domeniu

Domeniu nou: [Adaugă](#)

Adaugă autor

Autor nou: [Adaugă](#)

Adaugă carte

Domeniu:
 Autor:
 Isaac Asimov
 Alexandru Mitru
 Frank Herbert
 George Cosbuc
 Mihai Eminescu
 William Shakespeare

[Done](#)

Și lista autorilor din baza de date poate fi generată automat.

```
/* Afisăm și lista dropdown cu autori */
<?
$sql = "SELECT * FROM autori ORDER
        BY nume_autor ASC";
$resursa = mysql_query($sql);
while($row =
    mysql_fetch_array($resursa))
{
    print '<option
    value="'. $row['id_autor'].'">'.
    $row['nume_autor'].'</option>';
}
?>
</select>
```

```
</td>
</tr>
<tr>
<td>Titlu:</td>
<td><input type="text" name="titlu">
</td>
</tr>
<tr>
<td align="top">Descriere:</td>
<td><textarea name="descriere"
rows="8"></textarea></td>
</tr>
<tr>
<td>Preț:</td>
<td><input type="text" name="pret">
</td>
</tr>
<tr>
<td></td>
<td><input type="submit" name=
"adauga_carte" value="Adaugă"></td>
</tr>
</table>
</form>
</body>
</html>
```

prelucrare_adaugare.php

```
<?
include("autorizare.php");
include("admin_top.php");
/* Dacă s-a trimis formularul pentru
adăugare de domeniu */
if(isset($_POST['adauga_domeniu'])) ~
{
    /* Înainte de a introduce noul nume de
    domeniu verificăm două lucruri: să nu fie
    gol și să nu existe deja în baza de date. În
    cazul în care numele de domeniu este gol
    sau se află deja în tabelul domenii, afișăm
    un mesaj de eroare și oprim execuția
    scriptului: */
    if($_POST['domeniu_nou'] == "") ~
    {
        print 'Trebuie să completezi
        numele de domeniu!<br>
        <a href="adaugare.php">Înapoi</a>';
        exit;
    }
}
```

/* Verificăm dacă nu există deja în baza de date: */

```
$sql = "SELECT * FROM domenii WHERE
        nume_domeniu='".$_POST['domeniu_nou']."'"; ~
$resursa = mysql_query($sql);
```

/* Interrogarea returnează 0 rânduri
dacă domeniul nu există în baza de date.
Dacă nu returnează 0 înseamnă că dome-

niul specificat există deja în baza de date. Nu îl vom adăuga din nou ci doar vom atenționa utilizatorul de eroare și vom întrerupe execuția ulterioară a scriptului: */

```
if(mysql_num_rows($resursa) != 0)
{
    print 'Domeniul
    <b>'.$_POST['domeniu_nou']. '</b>
    există deja în baza de date!<br>
    <a href="adaugare.php">Înapoi</a>';
    exit;
}
```

/* Am verificat să nu fie erori, putem
acum să adăugăm noul nume de domeniu
în baza de date: */

```
$sql = "INSERT INTO domenii
        (nume_domeniu) VALUES
        ('".$_POST['domeniu_nou']."')";
mysql_query($sql);
/* și afișăm utilizatorului un mesaj de
confirmare: */
```

```
print 'Domeniul
    <b>'.$_POST['domeniu_nou']. '</b> a
    fost adăugat în baza de date!<br>
    <a href="adaugare.php">Înapoi</a>';
exit;
}
```

/* Același script, cu mici diferențe, îl
vom folosi pentru adăugarea unui autor
nou, dacă s-a trimis formularul de
adăugare autor: */

```
if(isset($_POST['adauga_autor']))
{
    /* Verificăm dacă nu este gol: */
    if($_POST['autor_nou'] == "") ~
    {

```

```
        print 'Trebuie să completezi numele
        autorului!<br>
        <a href="adaugare.php">Înapoi</a>';
        exit;
    }
}
```

/* Verificăm dacă nu există deja în
baza de date: */

Domeniul SF există deja în baza de date!

Dacă se încearcă adăugarea unui domeniu deja existent primim acest mesaj

CHIP SPECIAL - SITE DINAMIC

```

$sql = "SELECT * FROM autori WHERE
    nume_autor='".$POST['autor_nou']."' ";
$resursa = mysql_query($sql);
if(mysql_num_rows($resursa) != 0)
{
    print 'Autorul
        <b>' . $POST['autor_nou'] . '</b>
        există deja în baza de date!<br>
        <a href="adaugare.php">înapoi</a>';
    exit;
}

/* Am verificat să nu fie erori, putem
acum să adăugăm noul autor în baza de
date: */

$sql = "INSERT INTO autori(nume_autor)
VALUES ('" . $POST['autor_nou'] . "')";
mysql_query($sql);

/* și afișăm utilizatorului un mesaj de
confirmare: */

print 'Autorul
        <b>' . $POST['autor_nou'] . '</b> a
        fost adăugat în baza de date!<br>
        <a href="adaugare.php">înapoi</a>';
exit;

*/

```

/* Scriptul pentru adăugarea cărții va fi o idee mai complicată deoarece avem mai multe variabile de verificat, dar în mare are același mod de funcționare: */

```

if(isset($_POST['adauga_carte']))
{

```

Formularul cu ajutorul căruia putem adăuga rapid o carte.

/* Verificăm dacă titlul, descrierea sau prețul nu sunt goale: */

```

if($POST['titlu'] == "" ||
$POST['descriere'] == "" ||
$POST['pret'] == "")
{
    print 'Titlul, descrierea sau
    prețul sunt goale!<br>
    <a href="adaugare.php">înapoi</a>';
    exit;
}

/* Verificăm dacă valoarea introdusă
în câmpul Preț este de tip numeric folosind funcția is_numeric: */

if(!is_numeric($POST['pret']))
{
    print 'Câmpul Preț trebuie să fie
    de tip numeric! (scrieți <b>1000
    </b> în loc de <b>1000 lei</b>)<br>
    <a href="adaugare.php">înapoi</a>';
    exit;
}

```

/* Verificăm dacă această carte nu există deja în baza de date, după două criterii: id_autor și titlu. */

```

$sql = "SELECT * FROM carti WHERE
    id_autor='".$POST['id_autor']."' &&
    AND titlu='".$POST['titlu']."' ";
$resursa = mysql_query($sql);
if(mysql_num_rows($resursa) != 0)
{

```

```

    print 'Această carte există deja în
    baza de date!<br>
    <a href="adaugare.php">înapoi</a>';
    exit;
}

```

/* Am verificat să nu fie erori, putem acum să adăugăm carte în baza de date. */

```

$sql = "INSERT INTO carti(id_domeniu,
    id_autor, titlu, descriere, pret)
VALUES ('" . $POST['id_domeniu'] . "','" .
$POST['id_autor'] . "','" . $POST['titlu'] . "','" .
$POST['descriere'] . "','" . $POST['pret'] . "')";
mysql_query($sql);

/* și afișăm utilizatorului un mesaj de
confirmare: */

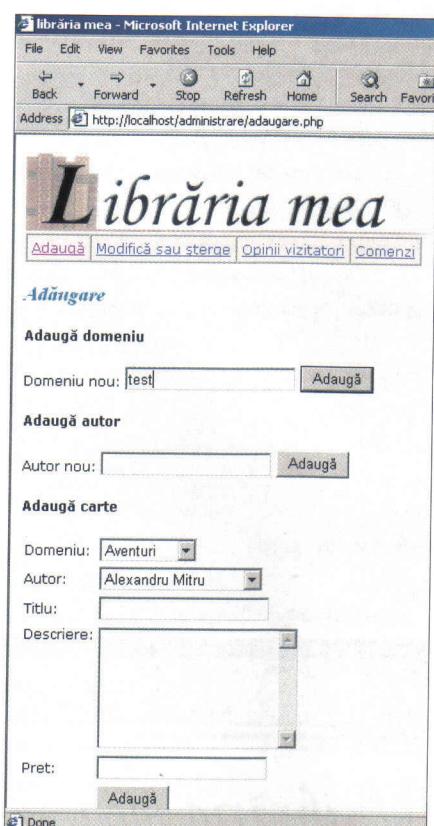

```

```

print 'Cartea a fost adăugată în baza
de date!<br>
<a href="adaugare.php">înapoi</a>';
exit;
}
?>
</body>
</html>

```

Rulați aceste scripturi pentru a introduce cărți, autori și domenii noi în baza de date. Accesați prima pagină a site-



Adăugarea unui domeniu nou.

ului, http://localhost pentru a vedea adăugările, conținutul nou al site-ului, adăugat în câteva secunde!

Modificare și stergere

Ce ne facem atunci când am făcut o greșală scriind greșit un nume de domeniu sau titlul unei cărți?

Dar dacă nu mai avem o carte în stoc și dorim să o scoatem de pe site?

Vom scrie trei fișiere pentru modificarea sau stergerea de elemente ale site-ului nostru.

Fișierul `modificare_stergere.php` va conține trei formulare: unul pentru numele de domenii, unul pentru numele autorilor și altul pentru cărți.

Numele de domenii și numele autorilor le vom afișa în liste dropdown, astfel încât să putem selecta imediat domeniul sau autorul ce dorim să îl stergem.

Formularul pentru cărți va conține două câmpuri: autor și titlu. Numele autorului va fi disponibil într-o listă dropdown, iar titlul cărții îl vom scrie într-un `<input type="text">`.

Iată codul acestui script și puteți vedea în imaginea de la pag. 58 rezultatul său:

modificare_stergere.php

```
<?
include("autorizare.php");
include("admin_top.php");
?>
<h1>Modificare sau ștergere</h1>
<p><b>Notă:</b> Nu veți putea
șterge domenii care au cărți în
ele. Înainte de a șterge domeniul,
modificați cărțile din el astfel
încât să aparțină altor domenii.
De asemenea nu veți putea șterge
un autor dacă există cărți în baza
de date care au acel autor.</p>
<b>Selectează domeniul ce dorești
să îl modifici sau ștergi:</b>
<form
action="formulare_modificare_stergere.php"
method="POST">
Domeniu:
<select name="id_domeniu">
/* Luăm numele de domenii din baza
de date și le afișăm utilizatorului într-o
listă drop-down. Astfel putem obține un
id_domeniu corespunzător domeniului
selectat pe care să îl introducem în tabele
carti */
<?
$sql = "SELECT * FROM domenii ORDER
BY nume_domeniu ASC";
$resursa = mysql_query($sql);
while($row =
mysql_fetch_array($resursa))
{
print '<option
value="'. $row['id_domeniu'].'">' .
$row['nume_domeniu']. '</option>';
}
?>
</select>
<INPUT type="submit" name=
"modifica_domeniu" value="Modifică">
<INPUT type="submit" name=
"sterge_domeniu" value="Șterge">
</form>

<b>Selectează autorul ce dorești să
il modifici sau ștergi:</b>
<form
action="formulare_modificare_stergere.php"
method="POST">
Autor:
<select name="id_autor">
/* Afisăm și lista dropdown cu autori */
<?
$sql = "SELECT * FROM autori ORDER
BY nume_autor ASC";
$resursa = mysql_query($sql);
while($row =
mysql_fetch_array($resursa))
{
print '<option
value="'. $row['id_autor'].'">' .
$row['nume_autor']. '</option>';
}
?>
</select>
<INPUT type="submit" name=
"modifica_carte" value="Modifică">
<INPUT type="submit" name=
"sterge_carte" value="Șterge">
</form>
</body>
</html>
```

```
$resursa = mysql_query($sql);
while($row =
mysql_fetch_array($resursa))
{
print '<option
value="'. $row['id_autor'].'">' .
$row['nume_autor']. '</option>';
}
?>
</select>
<INPUT type="submit" name=
"modifica_carte" value="Modifică">
<INPUT type="submit" name=
"sterge_carte" value="Șterge">
</form>
</body>
</html>
```

Fișierul formulare_modificare_stergere.php va prelua informațiile din modificare_

stergere.php și va afișa câte un formular pentru fiecare caz în parte:

- dacă a fost apăsat butonul de modificare nume domeniu, va afișa numele de domeniu într-un textbox astfel încât să îl putem modifica;

- dacă a fost apăsat butonul de ștergere domeniu, va verifica întâi dacă sunt cărți în domeniu respectiv. Dacă sunt, va afișa lista acestora și va afișa mesajul „Acest domeniu nu poate fi șters deoarece sunt cărți care îi aparțin!”. Dacă nu este nici o carte în domeniu respectiv, va afișa un mesaj cu o cerere de confirmare și un buton pe care să apăsăm pentru a confirma ștergerea.

- similar, dacă a fost apăsat butonul de modificare nume autor, va afișa numele autorului într-un textbox astfel încât să îl putem modifica;

- dacă a fost apăsat butonul de ștergere autor, va verifica întâi dacă sunt cărți în baza de date care corespund autorului. Dacă sunt, va afișa lista acestora și va afișa mesajul „Acest autor nu poate fi șters deoarece sunt cărți în baza de date care îi aparțin!”. Dacă nu este nici o carte scrisă de autor în baza de date, va afișa un mesaj cu o cerere de confirmare și un buton pe care să apăsăm pentru a confirma ștergerea.

- atunci când se apasă butonul de modificare carte, verifică întâi dacă există o carte în baza de date care are titlul și autorul mentionate (titlul este scris de către utilizator și trebuie să ne asigurăm că l-a scris corect). Dacă nu există nici o carte cu datele menționate afișează un mesaj de eroare. Dacă în schimb cartea există, afișează un formular cu ajutorul căruia să putem modifica domeniul, autorul, titlul, descrierea și prețul cărții.

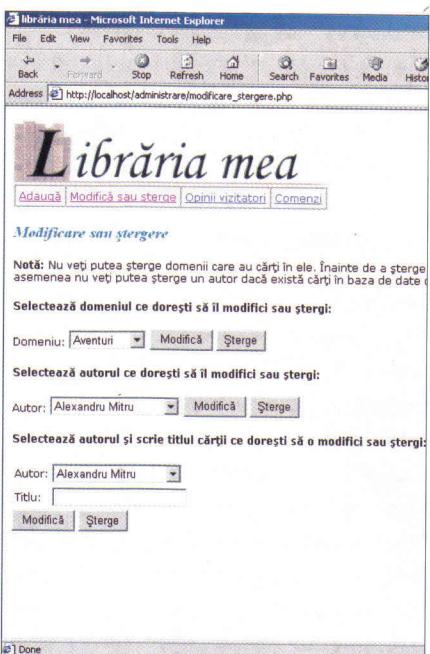
- dacă s-a apăsat butonul de ștergere carte, verifică dacă există o carte cu titlul și autorul menționate. Dacă nu există afișează un mesaj de eroare iar dacă există afișează un mesaj de confirmare și un buton pe care să apăsăm pentru a confirma ștergerea.

formulare_modificare_stergere.php

```
<?
include("autorizare.php");
include("admin_top.php");
/* modificare nume domeniu */
if(isset($_POST['modifica_domeniu']))
{
/* luăm numele de domeniu din baza
```

de date deoarece ne-a fost trimis din formular doar id-ul domeniului: */

```
$sql = "SELECT nume_domeniu FROM
domenii WHERE
id_domeniu='".$_POST['id_domeniu']."' ";
$resursa = mysql_query($sql);
$nume_domeniu =
mysql_result($resursa, 0,
"nume_domeniu");
/* și afișăm numele vechi de domeniu
într-un textbox pentru a fi modificat */
?>
<h1>Modifică nume domeniu</h1>
<form action=
"prelucrare_modificare_stergere.php"
method="POST">
<INPUT type="text"
name="nume_domeniu"
value=<?=$nume_domeniu?>">
<INPUT type="hidden"
name="id_domeniu"
value=<?=$_POST['id_domeniu']?>">
<INPUT type="submit"
name="modifica_domeniu"
value="Modifică">
</form>
<?
/* stergere domeniu */
if(isset($_POST['sterge_domeniu']))
{
/* verificăm dacă sunt cărți în baza de
date care aparțin acestui domeniu: */
$sql = "SELECT titlu, nume_autor
FROM carti, autori, domenii WHERE
carti.id_domeniu=domenii.id_domeniu
AND carti.id_autor=autori.id_autor
AND domenii.id_domeniu=
".$_POST['id_domeniu'];
$resursa = mysql_query($sql);
$nrCarti = mysql_num_rows($resursa);
/* dacă sunt cărți aparținând acestui
domeniu, afișăm lista lor și un mesaj de
eroare: */
if($nrCarti > 0)
{
print "<p>Sunt $nrCarti cărți
care aparțin acestui domeniu!</p>";
while($row =
mysql_fetch_array($resursa))
{
print "<b>".$row['titlu']."</b>
de ".$row['nume_autor']."<br>";
}
print "<p>Nu puteți sterge acest
domeniu!</p>";
}
}
else
{
?>
<h1>Sterge nume domeniu</h1>
Ești sigur că vrei să stergi ac-
est domeniu?
<form
action="prelucrare_modificare_stergere.php"
method="POST">
<INPUT type="hidden" name="id_domeniu"
value=<?=$_POST['id_domeniu']?>">
<INPUT type="submit" name=
"sterge_domeniu" value="Sterge!">
</form>
<?
}
/* modificare nume autor */
if(isset($_POST['modifica_autor']))
{
/* luăm numele autorului din baza de
date deoarece ne-a fost trimis din formu-
lar doar id_autor: */
$sql = "SELECT nume_autor FROM
autori WHERE id_autor=
'".$_POST['id_autor']."' ";
$resursa = mysql_query($sql);
$nume_autor = mysql_result
($resursa, 0, "nume_autor");
/* și afișăm numele într-un textbox
pentru a fi modificat */
?>
```



Pagina cu ajutorul căreia vom modifi-
ca sau șterge informații de pe site.

/* iar dacă nu sunt cărți în acest dome-
niu cerem confirmarea pentru ștergere:
*/

```
else
{
?>
<h1>Sterge nume domeniu</h1>
Ești sigur că vrei să stergi ac-
est domeniu?
<form
action="prelucrare_modificare_stergere.php"
method="POST">
<INPUT type="hidden" name="id_domeniu"
value=<?=$_POST['id_domeniu']?>">
<INPUT type="submit" name=
"sterge_domeniu" value="Sterge!">
</form>
<?
}
```

/* modificare nume autor */
if(isset(\$_POST['modifica_autor']))
{
/* luăm numele autorului din baza de
date deoarece ne-a fost trimis din formu-
lar doar id_autor: */
\$sql = "SELECT nume_autor FROM
autori WHERE id_autor=
'".\$_POST['id_autor']."' ";
\$resursa = mysql_query(\$sql);
\$nume_autor = mysql_result
(\$resursa, 0, "nume_autor");
/* și afișăm numele într-un textbox
pentru a fi modificat */
?>

```
<h1>Modifică nume autor</h1>
<form
action="prelucrare_modificare_stergere.php"
method="POST">
<INPUT type="text"
name="nume_autor"
value=<?=$nume_autor?>">
<INPUT type="hidden"
name="id_autor"
value=<?=$_POST['id_autor']?>">
<INPUT type="submit"
name="modifica_autor"
value="Modifică">
</form>
<?
}
/* ștergere autor */
if(isset($_POST['sterge_autor']))
{
/* verificăm dacă sunt cărți în baza de
date care aparțin acestui autor: */

```

```
$sql = "SELECT titlu FROM carti,
autori WHERE carti.id_autor=
autori.id_autor AND
carti.id_domeniu=
".$_POST['id_domeniu'];
$resursa = mysql_query($sql);
$nrCarti = mysql_num_rows($resursa);
/* dacă sunt cărți aparținând acestui
autor, afișăm lista lor și un mesaj de
eroare: */
if($nrCarti > 0)
{
print "<p>Sunt $nrCarti cărți de
acest autor în baza de date!</p>";
while($row =
mysql_fetch_array($resursa))
{
print $row['titlu']."<br>";
}
print "<p>Nu puteți șterge acest
autor!</p>";
}
/* iar dacă nu sunt cărți de acest autor
cerem confirmarea pentru ștergere: */
else
{
?>
```

```
<h1>Sterge autor</h1>
Ești sigur că vrei să stergi
acest autor?
<form
action="prelucrare_modificare_stergere.php"
method="POST">
<INPUT type="hidden"
name="id_autor"
value=<?=$_POST['id_autor']?>">
```

CHIP SPECIAL - SITE DINAMIC

```

<INPUT type="submit"
       name="sterge_autor"
       value="$sterge!">
</form>
<?
}
/* modificare carte */
if(isset($_POST['modifica_carte']))
{
print "<h1>Modificare carte</h1>";
/* căutăm întâi o carte în baza de date care
   are titlul și id_autor specificate în formular: */
$sqlCarte = "SELECT * FROM carti
WHERE titlu='".$_POST['titlu']."'"
AND id_autor='".$_POST['id_autor'];";
$resursaCarte = mysql_query($sqlCarte);
/* dacă nu s-a găsit nici o carte care să
   corespundă datelor introduse, afișăm un
   mesaj de eroare: */
if(mysql_num_rows($resursaCarte) == 0)
{
print "Această carte nu există în
      baza de date";
}
else
{

```

/* dacă există, atunci extragem
 informațiile din resursă, le punem într-un
 array (nu folosim while deoarece este
 returnat un singur rând!) și le afișăm în
 formular pentru a fi modificate: */

```

$rowCarte =
mysql_fetch_array($resursaCarte);
?>
<form
action="prelucrare_modificare_stergere.php"
method="POST">
<table>
<tr>
<td>Domeniu:</td>
<td>
<select name="id_domeniu">
<?
```

/* Luăm numele de domenii din baza
 de date și le afișăm utilizatorului într-o
 listă drop-down. Observați folosirea lui
 if pentru a afișa ca selectat domeniul de
 care aparține cartea: */

```

$sql = "SELECT * FROM domenii
ORDER BY nume_domeniu ASC";
$resursa = mysql_query($sql);
while($row =
mysql_fetch_array($resursa))
{

```

```

if($row['id_domeniu'] ==
$rowCarte['id_domeniu'])
{
print '<option SELECTED
value="'.$row['id_domeniu'].'">' .
$row['nume_domeniu'] .
'</option>';
}
else
{
print '<option
value="'.$row['id_domeniu'].'">' .
$row['nume_domeniu'] .
'</option>';
}
?>
</select>
</td>
</tr>
<tr>
<td>Autor:</td>
<td>
<select name="id_autor">
<?
/* Afișăm și lista dropdown cu autori
*/
$sql = "SELECT * FROM autori
ORDER BY nume_autor ASC";
$resursa = mysql_query($sql);
while($row =
mysql_fetch_array($resursa))
{
if($row['id_autor'] ==
$rowCarte['id_autor'])
{
print '<option SELECTED
value="'.$row['id_autor'].'">' .
$row['nume_autor'] .
'</option>';
}
else
{
print '<option
value="'.$row['id_autor'].'">' .
$row['nume_autor'] .
'</option>';
}
?>
</select>
</td>
</tr>
<tr>
<td>Titlu:</td>
<td>
<INPUT type="text" name="titlu"
```

```

value="<?=$rowCarte['titlu']?>">
</td>
</tr>
<tr>
<td valign="top">Descriere:</td>
<td><textarea name="descriere"
rows="8"><?=$rowCarte['descriere']?>
</textarea></td>
</tr>
<tr>
<td>Pret:</td>
<td><INPUT type="text" name="pret"
value="<?=$rowCarte['pret']?>">
</td></tr>
</table>
<INPUT type="hidden"
name="id_carte"
value="<?=$rowCarte['id_carte']?>">
<INPUT type="submit"
name="modifica_carte"
value="Modifică">
</form>
<?
}
}
```

/* și în final stergere carte */
if(isset(\$_POST['sterge_carte']))
{

```

print "<h1>Sterge carte</h1>";
/* căutăm întâi o carte în baza de date
   care are titlul și id_autor specificate în
   formular: */
$sqlCarte = "SELECT * FROM carti
WHERE titlu='".$_POST['titlu']."'"
AND id_autor='".$_POST['id_autor'];";
$resursaCarte=mysql_query($sqlCarte);
/* dacă nu s-a găsit nici o carte care să
   corespundă datelor introduse, afișăm un
   mesaj de eroare: */
if(mysql_num_rows($resursaCarte)==0)
{
print "Această carte nu există în
      baza de date";
}
```

/* iar dacă există, atunci extragem id-ul
 cărții din baza de date și îl vom folosi
 într-un câmp ascuns din formularul de
 confirmare: */

```

else
{
$id_carte = mysql_result
($resursaCarte,0,id_carte");
?>
Ești sigur că vrei să stergi
această carte?
<form
action="prelucrare_modificare_stergere.php"
```



Formularul pentru modificarea unui nume de domeniu scris greșit.

```

method="POST">
<INPUT type="hidden"
       name="id_carte"
       value="=$id_carte?"&gt;
&lt;INPUT type="submit"
       name="sterge_carte"
       value="Șterge!"&gt;
&lt;/form&gt;
&lt;?
}
?&gt;
&lt;/body&gt;
&lt;/html&gt;
</pre

```

După cum vedem, pagina formulare_modificare_stergere.php conține doar formulare, pentru toate operațiunile de modificare și ștergere din baza de date. Să mai facem un ultim fișier, în care să prelucrăm datele trimise prin aceste formulare. Structura sa va fi următoarea:

- modificarea numelui de domeniu în



Modificarea informațiilor unei înregistrări este simplă.

baza de date;

- (sau) ștergerea domeniului din baza de date;

- (sau) modificarea numelui autorului ;

- (sau) ștergerea autorului;

- (sau) actualizarea informațiilor despre carte;

- (sau) ștergerea cărții din baza de date și a tuturor comentariilor utilizatorilor la cartea respectivă.

Codul este următorul (semnificativ mai simplu decât cel al fișierului precedent):

prelucrare_modificare_stergere.php

```

<?
include("autorizare.php");
include("admin_top.php");
/* modificare nume domeniu */
if(isset($_POST['modifica_domeniu']))
{
/* Verificăm dacă noul nume de domeniu a fost introdus. */
if($_POST['nume_domeniu'] == "")
{
print "Nu ați introdus numele domeniului!";
}
else
{
$sql = "UPDATE domenii SET
        nume_domeniu='".$POST['nume_domeniu']."'.
        WHERE id_domeniu=".
        $_POST['id_domeniu'];
mysql_query($sql);
print "Numele domeniului a fost
        modificat!";
}

/* De ce nu am folosi exit în structura if(condiție){codul de executat; exit;}?
Pentru că dacă nu se execută codul din if(conditie){}, se execută codul din else{} și atunci exit ar fi superfluu. */
}

/* ștergere domeniu */
if(isset($_POST['sterge_domeniu']))
{
$sql = "DELETE FROM domenii WHERE
        id_domeniu=".$POST['id_domeniu'];
mysql_query($sql);
print "Domeniul a fost sters!";
}

/* modificare nume autor */
if(isset($_POST['modifica_autor']))
{
/* Verificăm dacă numele modificat al autorului a fost introdus. */
}

```

```

if($_POST['nume_autor'] == "")
{
print "Nu ați introdus numele autorului!";
}
else
{
$sql="UPDATE autori SET nume_autor=
      '".$POST['nume_autor']."' WHERE
      id_autor=".$POST['id_autor'];
mysql_query($sql);
print "Numele autorului a fost
      modificat!";
}
/* ștergere autor */
if(isset($_POST['sterge_autor']))
{
$sql = "DELETE FROM autori WHERE
        id_autor=".$POST['id_autor'];
mysql_query($sql);
print "Autorul a fost sters!";
}
/* modificare informații carte */
if(isset($_POST['modifica_carte']))
{

```

/* Verificăm dacă toate datele au fost introduce corect. N-am vrea să introducem date eronate în baza de date doar pentru că a sărit pisica pe tastatură și a apăsat ENTER în timp ce introduceam datele. Dacă credeți că nu vi se poate întâmpla... ei bine, din proprie experiență vă spun că se poate. Vom folosi o structură if ... else if... else: */

```

if($_POST['titlu'] == "")
{
print "Nu ați introdus titlul!";
}
else if($_POST['descriere'] == "")
{
print "Nu ați introdus descrierea!";
}

```



Nu putem șterge un domeniu în care există cărți.

```

else if($_POST['pret'] == "")
{
    print "Nu ati introdus pretul!";
}

else if(!is_numeric($_POST['pret']))
{
    print "Prețul trebuie să fie
        numeric! Scrieți <b>1000</b>,
        nu <b>1000 lei</b>!";
}

else
{
    $sql = "UPDATE carti SET
        id_domeniu=".$_POST['id_domeniu'],
        id_autor=".$_POST['id_autor'],
        titlu='".$_POST['titlu'],
        descriere='".$_POST['descriere'],
        pret=".$_POST['pret']."' WHERE
        id_carte=".$_POST['id_carte'];
    mysql_query($sql);
    print "Informațiile au fost
        modificate!";
}

/* ștergere carte */
if(isset($_POST['sterge_carte']))
{
    $sqlCarte = "DELETE FROM carti
        WHERE id_carte="
        .$_POST['id_carte'];
    mysql_query($sqlCarte);

    $sqlComentarii="DELETE FROM comentarii
        WHERE id_carte=".$_POST['id_carte'];
    mysql_query($sqlComentarii);
    print "Cartea a fost stearsă din
        baza de date!";
}
?>
</body>
</html>

```

Moderare comentarii

Următorul lucru de care ne vom ocupa va fi moderarea comentariilor vizitatorilor. Deși utilizatorul obișnuit este bine cresut și cunoaște regulile netichetei nu toți sunt așa și există posibilitatea să trebuiască să editați sau chiar să ștergeți unele comentarii.

Să monitorizăm fiecare pagină de carte în fiecare zi pentru a vedea ce comentarii noi au fost posteate nu este o soluție viabilă: necesită mult timp și ar fi practic imposibilă în cazul unui site cu multe cărți. De aceea, vom avea în zona de

administrare o secțiune specială pentru moderare unde vom afișa comentariile noi (adăugate de la ultima moderare).

Pentru a afla care sunt comentariile noi de la ultima accesare a secțiunii de moderare va trebui să specificăm undeva care este ultimul comentariu gata moderat. Nu vom seta un cookie pentru că atunci am fi condiționați de folosirea același calculator și browser pentru accesarea paginii de administrare. Vom adăuga încă un câmp în tabelul admin unde vom păstra id-ul ultimului comentariu moderat.

Cum id-urile sunt create în ordine crescătoare, comentariile noi vor fi cele care au id mai mare decât cel setat în tabelul admin. Să creăm noul câmp:

```

ALTER TABLE admin ADD
    ultimul_comentariu_moderat INT
    UNSIGNED DEFAULT "0" NOT NULL;

```

Ultimul comentariu moderat este în momentul de față 0. Comentariile noi le putem afla cu interogarea:

```

SELECT * FROM comentarii, admin
    WHERE comentarii.id_comentariu >
        admin.ultimul_comentariu_moderat;

```

Pentru a afla și titlurile și autorii cărților la care sunt comentarii noi, interogarea este:

```

SELECT * FROM comentarii, admin,
    carti, autori WHERE id_comentariu >
        admin.ultimul_comentariu_moderat AND
        carti.id_carte=comentarii.id_carte
        AND carti.id_autor=autori.id_autor;

```

Vom folosi această interogare în pagina de moderare comentarii și vom adăuga un formular cu ajutorul căruia să putem modifica valoarea câmpului `ultimul_comentariu_moderat` la fiecare vizită astfel încât dacă azi moderăm comentariile, mâine să nu vedem decât comentariile adăugare după ultimul comentariu moderat de azi. De asemenea, vom avea butoane pentru modificare sau ștergere comentariu.

După ce vom revizui comentariile utilizatorilor, modificându-le sau ștergându-le pe cele nepotrivate, vom putea apăsa butonul „Setează aceste comentarii ca fiind moderate” pentru a nu le mai vedea la următoarea verificare.

opinii.php

```

<?
include("autorizare.php");
include("admin_top.php");
?>

<h1>Modificare sau ștergere comentarii utilizatori</h1>
<b>Comentariile utilizatorilor de la
ultima moderare</b>
<?

$sql = "SELECT * FROM comentarii,
    admin, carti, autori WHERE
    id_comentariu>admin.ultimul_comentariu_
    moderat AND carti.id_carte=
    comentarii.id_carte AND
    carti.id_autor=autori.id_autor
    ORDER BY id_comentariu ASC";
$resursa = mysql_query($sql);
while($row =
    mysql_fetch_array($resursa))
{
?>
<form
action="formulare_moderare_opinii.php"
method="POST">
<div style="width:500px; border:1px
solid #fffff; background-
color:#F9F1E7; padding:5px">
<b><?=$row['titlu']?></b> de
<?=$row['nume_autor']?>
<HR size="1">
<a href=
    "mailto:<?=$row['adresa_email']?>">
<?=$row['nume_utilizator']?>
</a><br>
<?=$row['comentariu']?>
</div>

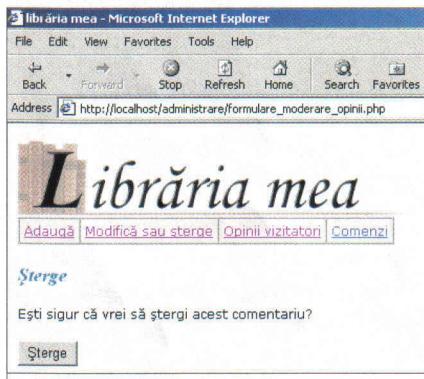
```



Comentariile noi și încă nemoderate ale utilizatorilor site-ului.



Formularul cu ajutorul căruia putem edita un comentariu nepotrivit...



...sau chiar șterge.

```
<INPUT type="hidden"
       name="id_comentariu"
       value="<?=$row['id_comentariu']?>">
<INPUT type="submit"
       name="modifica" value="Modifică">
<INPUT type="submit" name="sterge"
       value="Șterge">
</form>
?>
$ultimul_id = $row['id_comentariu'];
/* Această variabilă preia la fiecare
iterare cu while a array-ului valoarea
id_comentariu, ajungând ca la ultima ite-
rare să aibă valoarea id-ului ultimului
comentariu.
```

Aveam nevoie de această valoare astfel încât să putem seta valoarea câmpului `id_comentariu_moderat` din tabelul `admin` folosind formularul următor. */

```
?>
/* În continuare vom scrie o structură
condițională.
```

Dacă sunt comentarii în listă afișăm formularul și butonul de setare a comentariilor ca fiind moderate. Dacă nu e nici un comentariu în listă vom afișa doar un mesaj. Astfel evităm erorile care ar putea

apărea dacă nu avem comentarii în listă și valoarea variabilei `$ultimul_comentariu` ar fi nulă. */

```
$nrComentarii=mysql_num_rows($resursa);
if($nrComentarii > 0)
{
?>
<form
action="formulare_moderare_opinii.php"
method="POST">
<INPUT type="hidden"
       name="ultimul_id"
       value="<?=$ultimul_id?>">
<INPUT type="submit"
       name="seteaza_moderate"
       value="Setează aceste comentarii
ca fiind moderate">
</form>
?>
}
else
{
    print "<p>Nu există comentarii
noi.</p>";
}
?>
</body>
</html>
```

Fișierul care preia informațiile din `opinii.php` va afișa 3 formular, în funcție de butonul apăsat.

- dacă s-a apăsat butonul „Modifică” afișează un formular astfel încât să putem edita conținutul mesajului;

- dacă s-a apăsat butonul „Șterge” afișează un mesaj de confirmare și un buton pe care putem apăsa pentru a confirma ștergerea;

- dacă s-a apăsat butonul „Setează aceste comentarii ca fiind moderate”, afișează un mesaj și un buton pe care putem apăsa pentru a confirma că am terminat moderarea comentariilor din pagina precedentă.

```
formulare_moderare_opinii.php
?>
include("autorizare.php");
include("admin_top.php");
/* formular modificare comentariu*/
if(isset($_POST['modifica']))
{
    $sql = "SELECT * FROM comentarii WHERE
           id_comentariu=".$_POST['id_comentariu'];
    $resursa = mysql_query($sql);
```

/* fiind returnat un singur rând, nu folosim while */

```
$row = mysql_fetch_array($resursa);
?>
<h1>Modifică</h1>
<b>Modifică acest comentariu</b>
<form
action="prelucrare_moderare_comentarii.php"
method="POST">
    Nume: <input type="text"
               name="nume_utilizator"
               value="<?=$row['nume_utilizator']?>">
    Email: <input type="text"
                  name="adresa_email"
                  value="<?=$row['adresa_email']?>"><br><br>
    Comentariu: <br><textarea name=
"comentariu" cols="45" rows="8">
<?=$row['comentariu']?>
</textarea><br><br>
    <input type="hidden"
           name="id_comentariu"
           value="<?=$_POST['id_comentariu']?>">
    <input type="submit"
           name="modifica" value="Modifică">
</form>
?>
}
/* confirmare ștergere comentariu */
if(isset($_POST['sterge']))
{
?>
<h1>Șterge</h1>
Ești sigur că vrei să ștergi acest
comentariu?
<form
action="prelucrare_moderare_comentarii.php"
method="POST">
    <input type="hidden"
           name="id_comentariu"
           value="<?=$_POST['id_comentariu']?>">
    <input type="submit"
           name="sterge" value="Șterge">
</form>
?>
}
/* confirmare moderare */
if(isset($_POST['seteaza_moderate']))
{
?>
<h1>Setează comentariile ca fiind
moderate</h1>
Ești sigur că vrei să setezi comentariile din pagina precedentă ca fiind
moderate? Le-ai verificat pe toate?
<form
action="prelucrare_moderare_comentarii.php"
method="POST">
```



Comentariile deja moderate nu vor mai apărea la următoarea vizită în secțiunea administrativă.



Nu există nici un comentariu nou ne-moderat de la ultima accesare a secțiunii de administrare.

```
<input type="hidden"
       name="ultimul_id"
       value=<?=$_POST['ultimul_id']?>>
<input type="submit" name=""
       seteaza_moderate" value="Da!">
</form>
<?
}
?>
</body></html>
```

Și, în final, fișierul care preia datele din formulare_moderare_opinii.php și efectuează modificările în baza de date:

```
prelucrare_moderare_comentarii.php
<?
include("autorizare.php");
include("admin_top.php");
/* modificare comentariu*/
if(isset($_POST['modifica']))
{
    /* Verificăm să fie completeate câmpurile.*/
    if($_POST['nume_utilizator'] == "")
    {
        print "Nu ai completat numele
              utilizatorului!";
    }
    else if ($_POST['adresa_email'] == "")
    {
        print "Nu ai completat adresa de email!";
    }
    else if ($_POST['comentariu'] == "")
```

```

    {
        print "Câmpul Comentariu este gol!";
    }
    else
    {
        /* câmpurile sunt completate, să facem
           modificarea în baza de date*/
        $sql = "UPDATE comentarii SET
                nume_utilizator='".$_POST['nume_utilizator']."' ,
                adresa_email='".$_POST['adresa_email']."' ,
                comentariu='".$_POST['comentariu']."' ,
                WHERE id_comentariu=
                .$_POST['id_comentariu'];
        mysql_query($sql);
        print "Comentariul a fost modificat!";
    }
}

/* stergere comentariu */
if(isset($_POST['sterge']))
{
    $sql="DELETE FROM comentarii WHERE
          id_comentariu='".$_POST['id_comentariu'];
    mysql_query($sql);
    print "Comentariul a fost sters!";
}

/* setarea ultimului comentariu moderat în tabelul admin */
if(isset($_POST['seteaza_moderate']))
{
    $sql = "UPDATE admin SET
            ultimul_comentariu_moderat="
            .$_POST['ultimul_id'];
    mysql_query($sql);
    print "Valoarea a fost setată!";
}
?>
</body>
</html>
```

Testați funcționarea acestor scripturi, modificând și stergând comentarii. De acum, la fiecare nouă vizită în secțiunea de administrare nu veți vedea decât comentariile noi, nemoderate. Dacă mai adăugați administratori ai site-ului, toți vor vedea cele mai noi comentarii, indiferent de cine le-a moderat pe ultimele.

Comenzi

Am văzut cum atunci când se face o nouă comandă putem primi un email de notifi-

care. Ce ne facem însă dacă serverul nu poate trimite email? Sau dacă un virus șterge datele de pe partitura unde păstrăm emialurile? Putem face o pagină în secțiunea de administrare cu ajutorul căreia să monitorizăm comenziile, la fel cum moderăm comentariile. Avem o singură problemă: comenziile se fac în zile diferite, banii nu vin neapărat în ordinea în care avem comenziile în baza de date. Așadar nu putem pune pur și simplu un nou câmp în tabelul admin cu "ultima comandă onorată" deoarece pot exista comenzi mai vechi decât aceea, comenzi pentru care încă nu au venit banii. Va trebui atunci să facem o delimitare per comandă, indiferent de data sau id-ul acesteia. Vom face următorul lucru: în tabelul tranzactii (tabelul de comenzi) vom mai adăuga un câmp: comanda_onorata care poate avea două valori: 1 dacă tranzacția a fost efectuată cu succes (am primit banii, am trimis cărțile) sau 0 dacă comanda nu a fost (încă) onorată. Să adăugăm noul câmp în tabelul tranzactii, cu valoarea default 0:

```
ALTER TABLE tranzactii ADD
    comanda_onorata TINYINT UNSIGNED
    DEFAULT "0" NOT NULL;
SELECT * FROM tranzactii;
```

Să scriem și codul pentru fișierul comenzi.php. În acesta vom avea toate comenziile neonorate cu toate detaliile necesare: nume și adresă cumpărător, data, cărțile comandate, numărul și valoarea lor. De asemenea, pentru fiecare comandă vom avea două butoane: „Comandă onorată” pe care vom apăsa atunci când vom fi primit banii și trimis cărțile pentru comanda respectivă și „Anulează comanda” pe care vom apăsa atunci când comanda este foarte veche și utilizatorul nu a trimis încă banii sau atunci când utilizatorul ne anunță că nu mai dorește cărțile comandate). Iată codul:

```
comenzi.php
<?
include("autorizare.php");
include("admin_top.php");
?>
<h1>Comenzi</h1>
```

id_tranzactie	data_tranzactie	nume_cumparator	adresa_cumparator	comanda_onorata	
127	120030304	Ionescu Maria	Minei 13, Petrosani	0	(neonorată)
128	120030304	Georgescu Gheorghe	Medici nr. 17, Arad	1	(onorată)
129	120030304	Romeo Ciupercescu	Zorilor 30, Campina	0	(neonorată)

```

<b>Comenzi încă neonorate</b>
<?

/* Afisăm lista comenzilor neonorate
(WHERE comanda_onorata=0) din
tabelul tranzactii: */

$sqlTranzactii = "SELECT id_tranzactie,
DATE_FORMAT(data_tranzactie, '%d-%m-%Y')
as data_tranzactie,
nume_cumparator, adresa_cumparator
FROM tranzactii WHERE
comanda_onorata=0";

/* DATE_FORMAT este o funcție a
MySQL cu care putem forma o dată
stocată într-un timestamp după cum
dorim (în cazul de față, zz-ll-aaaa).
Funcția nu modifică nimic în baza de
date ci doar afișează un TIMESTAMP
într-un format mai ușor digerabil (04-
03-2003 în loc de 20030304). */

$resursaTranzactii =
mysql_query($sqlTranzactii);
while($rowTranzactie =
mysql_fetch_array($resursaTranzactii))
{
?>

<form
action="prelucrare_comenzi.php"
method="POST">

Data comenzi:
<b><?=$rowTranzactie['data_tranzactie']?>
</b>

<div style="width:500px; border:1px
solid #fffff; background-
color:#F9F1E7; padding:5px">
<b><?=$rowTranzactie['nume_cumparator']?>
</b><br>
<?=$rowTranzactie['adresa_cumparator']?>
<TABLE border="1" cellpadding="4"
cellspacing="0">
<tr>
<td align="center"><b>Carte</b></td>
<td align="center"><b>Nr. buc</b></td>
<td align="center"><b>Preț</b></td>
<td align="center"><b>Total</b></td>
</tr>
<?
/* și, pentru fiecare tranzactie, afișăm
cărțile comandate (titlul și autorul),
numărul și valoarea lor: */

$sqlCarti = "select titlu,
nume_autor, pret, nr_buc from
vanzari, carti, autori where
carti.id_carte=vanzari.id_carte
and carti.id_autor=autori.id_autor
and id_tranzactie=";
$rowTranzactie['id_tranzactie'];
$resursaCarti=mysql_query($sqlCarti);

```

```

while($rowCarte =
mysql_fetch_array($resursaCarti))
{
print '<tr><td>
'.$rowCarte['titlu'].' de
'.$rowCarte['nume_autor'].'.</td>
<td align="right">
'.$rowCarte['nr_buc'].'.</td>
<td align="right">
'.$rowCarte['pret'].'.</td>';
/* Calculăm totalul pentru această carte
(pret * nr_buc)*/
$total=$rowCarte['pret'] *
$rowCarte['nr_buc'];
/* Afisăm acest total și apoi îl adunăm
la totalul general pentru această comandă:
*/
print '<td
align="right">'. $total .'</td>
</tr>';
$totalGeneral = $totalGeneral + $total;
}
?>
<tr>
<TD colspan="3"
align="right">Total comandă:</td>
<td><?=$totalGeneral?> lei</td>
</tr>
</table>
<INPUT type="hidden"
name="id_tranzactie" value=
"?<?=$rowTranzactie['id_tranzactie']?>">
<INPUT type="submit"
name="comanda_onorata"
value="Comandă onorată">
<INPUT type="submit"
name="anuleaza_comanda"
value="Anulează comanda">
</div>
</form>
<?
?>
</body>
</html>

```

Nu vom mai trece prin formulare de confirmare (știți deja cum să faceți acest lucru), facem direct un fișier care operează modificările în baza de date:

```

prelucrare_comenzi.php
<?
include("autorizare.php");
include("admin_top.php");
print "<h1>Comenzi</h1>";

```

Carte	Nr. buc	Pret	Total
Preludul Fundației de Isaac Asimov	1	100000	100000
Fundata și Imperul de Isaac Asimov	1	121500	121500
			Total comandă: 221500 lei
<input type="button" value="Comandă onorată"/>		<input type="button" value="Anulează comanda"/>	

Listă comenzilor neonorate.

```

/* comanda a fost onorată*/
if(isset($_POST['comanda_onorata']))
{
/* setăm câmpul comanda_onorata=1
în tabelul tranzactii pentru această co-
mandă */
$sql = "UPDATE tranzactii SET
comanda_onorata=1 WHERE
id_tranzactie=".$_POST['id_tranzactie'];
mysql_query($sql);
/* și afișăm un mesaj*/
print "Comanda a fost onorată!";
}

/* comanda a fost anulată */
if(isset($_POST['anuleaza_comanda']))
{
/* ștergem tranzacția (din tabelul tranzactii) și cărțile comandate (din tabelul
vanzari)*/
$sqlTranzactii = "DELETE FROM
tranzactii WHERE id_tranzactie=".
$_POST['id_tranzactie'];
mysql_query($sqlTranzactii);
$sqlCarti = "DELETE FROM vanzari
WHERE id_tranzactie=".
$_POST['id_tranzactie'];
mysql_query($sqlCarti);
/* și afișăm un mesaj*/
print "Comanda a fost anulată!";
}
?>
</body>
</html>

```

Aceasta a fost secțiunea de administrare. Acum aveți un site funcțional, sigur, ușor de navigat și mai ales ușor de administrat.

Tips and tricks

Întrebări frecvente

În acest capitol veți afla cum să creați aplicații on-line sigure și cum să vă protejați de neplăcerile găurilor de securitate.

Am adunat în acest capitol răspunsurile la cele mai arzătoare întrebări puse de programatorii începători. Elemente care la prima vedere ar putea părea inofensive pot fi periculoase dacă sunt private din alt unghi. Variabilele globale, lipsa ghilimelelor magice și extensiile fișierelor pot fi folosite împotriva dumneavoastră.

Măsuri de securitate

Securizarea aplicațiilor on-line este un subiect vast abordat în nenumărate cărți. Aici vom vedea doar câteva din metodele pe care un programator PHP le are la îndemâna pentru a scrie aplicații mai sigure.

Regula numărul unu a securității on-line este: nu va încredeți niciodată în utilizator. Întotdeauna verificați datele trimise către server și „curățați-le” înainte de a le utiliza. Pentru aceasta trebuie să considerați câteva posibile „găuri”:

1. Variabilele globale

Având variabilele globale ON puteți accesa datele trimise prin formulare mai simplu: \$variabila în loc de \$_GET['variabila'], la fel și pentru POST sau FILES. Dacă scriptul nu este foarte bine gândit, variabilele globale pot prezenta un risc major de securitate. Din acest motiv php.ini este distribuit cu globals=off în ultimele versiuni.

2. Ghilimele „magice”

Dacă în php.ini magic_quotes_gpc sunt OFF folosiți funcția addslashes pentru a preceda ghilimelele din datele trimise de utilizatori cu caracterul \\. Dacă magic_quotes_gpc sunt ON, PHP adaugă automat caracterul \\ înainte de ghilimelele dar dacă ele sunt OFF, ghilimelele din input vă pot crea probleme serioase. Ca exemplu, să presupunem că interogarea SQL de verificare a numelui și parolei

pentru înregistrarea pe site este SELECT * FROM users WHERE nume='\$nume' and parola='\$parola' și dacă interogarea este executată cu succes, utilizatorul este logat. În acest caz, folosind parola ' OR '1 = 1 oricine poate avea acces pe site deoarece interogarea SELECT * FROM users WHERE nume='un nume oarecare' AND PAROLA=' OR '1 = 1' este executată cu succes și returnează un rezultat (toate înregistrările din baza de date, de fapt).

Mai mult, atunci când vă așteptați ca baza de date să returneze un singur rând, verificați acest lucru și nu dacă interogarea s-a executat cu succes:

```
/* aşa nu */
$sql = "SELECT * FROM users WHERE
nume='$nume' and parola='$parola'";
$resursa = mysql_query($sql);
if($resursa)
{
... autentificat...
}

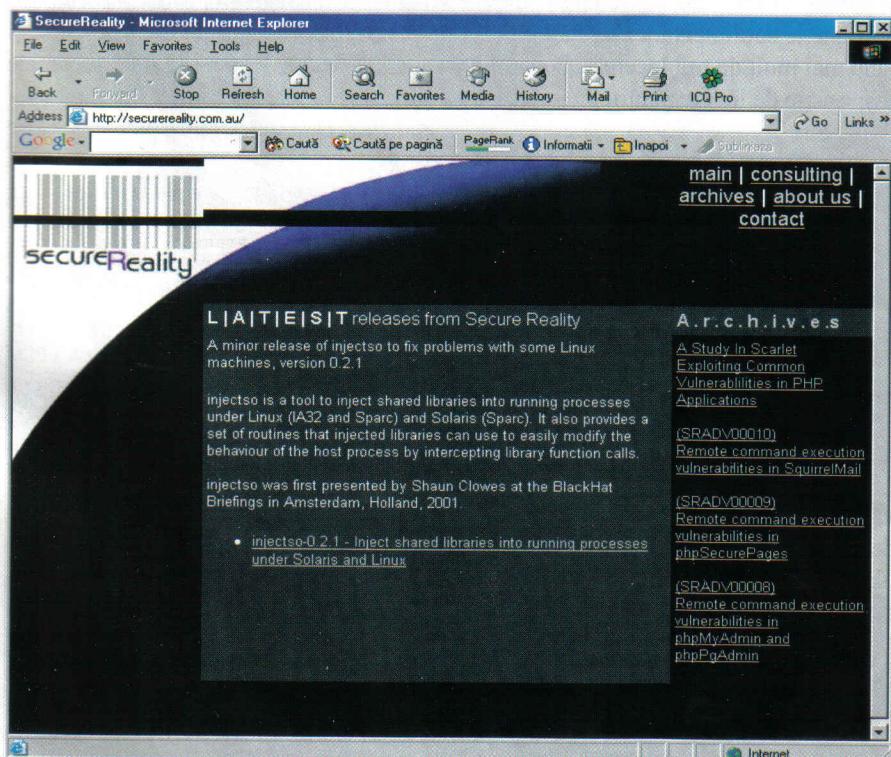
/* aşa da */
$sql = "SELECT * FROM users WHERE
nume='$nume' and parola='$parola'";
$resursa = mysql_query($sql);
if(mysql_num_rows($resursa) == 1)
{
... autentificat...
}
```

Notă: magic_quotes_gpc sunt implicit ON în php.ini pentru a vă proteja de astfel de atacuri, dar este bine să verificați înainte de a renunța la addslashes.

3. Includere

Încercați să evitați includerea „vizibilă” a fișierelor în forma http://site.ro/fisier.php?file=cutare.html pentru a include fișiere în cadrul unei pagini. Cu puțină neatenție din partea voastră atacatorul ar putea accesa astfel informații sensibile din cadrul sistemului.

Nu includeți fișiere străine. PHP poate „include” fișiere aflate pe alte servere decât cel care rulează dacă setarea **URL fopen wrappers** este activată în php.ini. În exemplul de mai sus, un atacator ar fi putut accesa adresa http://site.ro/



Pe site-ul Secure Reality găsiți mai multe articole privind securitatea PHP.

fisier.php?file=http://www.raul.ro/scriptrau.php pentru include în fișier un script localizat pe alt server și astfel obține acces către toate resursele sistemului la care are acces PHP, putând executa comenzi de sistem, afișa informații confidențiale sau șterge baza de date. Nu permiteți includerea fișierelor din altă parte decât de pe serverul vostru. Setați allow_url_fopen=Off în php.ini.

4. Formulare

Folosiți metoda POST în formulare atunci când informația din acestea urmează să fie introdusă în baza de date. Am să vă dau un exemplu: să presupunem că avem un formular cu ajutorul căruia utilizatorii pot introduce părerile lor despre site, formular care este prelucrat de către un fisier userinput.php:

[formular.html](#)

```
<form action="userinput.php"
method="GET">
<textarea name="parerea_mea">
</textarea>
<input type="submit">
</form>
```

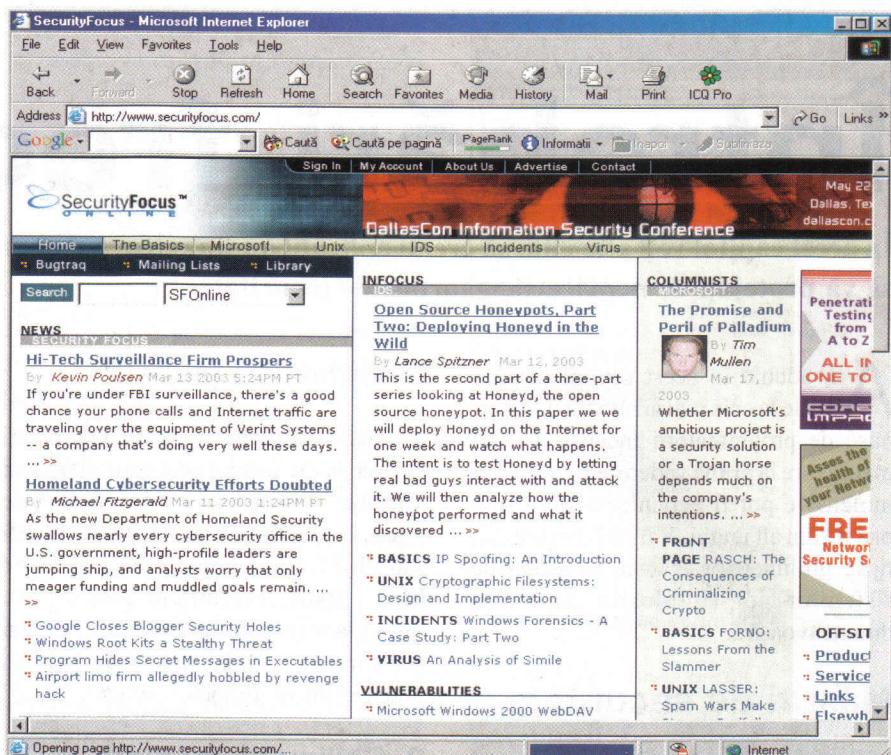
[userinput.php](#)

```
<?
mysql_connect(...);
mysql_query("insert into pareri
values ('$parerea_mea')");
?>
```

Dacă variabilele globale sunt OFF în php.ini sau metoda de transmitere a formularului este GET, un utilizator rău intenționat ar putea accesa adresa <http://site.ro/userinput.php?parereamea=blabla> și ar introduce comentariul „blabla” în baza voastră de date fără să treacă printrui-zis prin site.

Dacă vă gândiți cumva că asta n-ar fi chiar o problema atât de mare, imaginați-vă că „răufacatorul” scrie un script care accesează adresa fără oprire timp de câteva ore. Dacă legătura lui la Internet e bună, e posibil ca a doua zi dimineață să vă treziți fără spațiu pe server, cu câteva milioane de comentarii care spun același lucru: „blabla”.

Verificarea provenientei cererilor către server este foarte importantă și în alt caz: formularele de login. Cineva care vă știe numele de utilizator ar putea să încerce (și, cu ceva noroc, chiar să reușească) să vă găsească parola foarte ușor



Pe [SecurityFocus.com](http://www.securityfocus.com/) sunt prezentate cele mai noi găuri de securitate întâlnite în aplicații.

În acest mod. Având la dispoziție un dicționar, intuiția sau un generator de parole se poate accesa foarte ușor și rapid adresa www.site.ro/login.php?username=NumeleTau&parola=aaa, www.site.ro/login.php?username=tu&parola=aab, www.site.ro/login.php?username=tu&parola=aac și aşa mai departe până la descoperirea combinației valide de nume și parolă. În acest caz va trebui să puneti o protecție suplimentară care să nu permită mai mult de trei încercări consecutive eşuate de logare pentru un nume de utilizator.

Această problemă se rezolvă rapid folosind sesiunile. În momentul în care un utilizator trimite numele și parola către login.php, putem seta o variabilă de sesiune `$_SESSION['login_count']` care să țină minte numărul încercărilor. Când valoarea acesteia trece de 3 (trei încercări nereușite) nici măcar nu mai interogați baza de date pentru a verifica a patra încercare.

Variabila `$_SESSION['login_count']` va rămâne în memorie cât browserul este deschis și sesiunea activă (opțiunea implicită a PHP de menținere a sesiunilor active este de o oră). Dacă atacatorul așteaptă o oră sau își inchide browserul, sesiunea va fi închisă și va putea încerca de alte trei ori să se logheze.

De cele mai multe ori această masură de siguranță este suficientă pentru a preveni încercările de aflare a parolelor uti-

lizerilor. Închiderea și deschiderea browserului pentru a încerca de 3 ori este suficient de descurajantă ca să determine un hacker să își caute de lucru în altă parte.

5. Extensiile

O practică obișnuită este de a acorda extensia .inc fișierelor care conțin biblioteci de funcții ce urmează a fi incluse și folosite în cod. PHP nu parsează fișierele cu extensia .inc și dacă acestea sunt apelate direct ele sunt trimise plain text către browser.

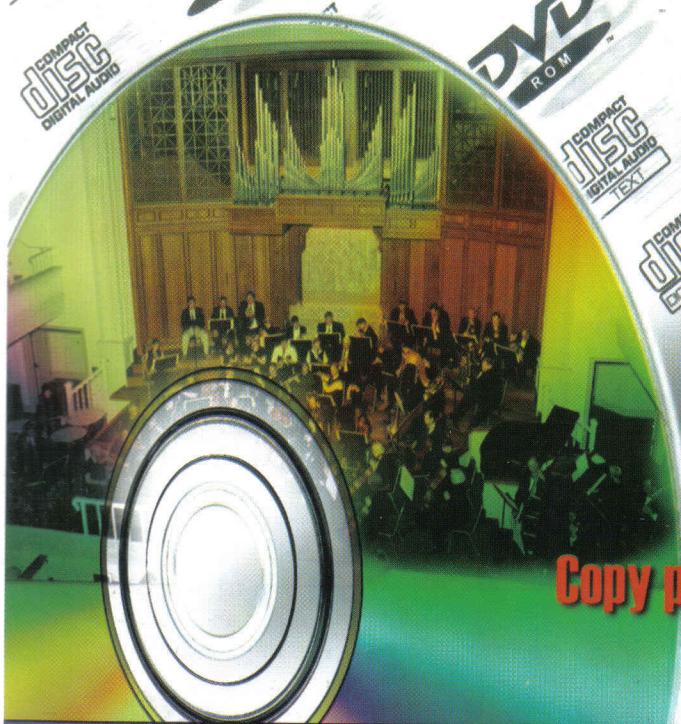
Evități expunerea informațiilor dacă dați acestor fișiere extensia .php - astfel ele vor fi rulate în loc să fie afișate. Nu puneti informații sensibile (precum numele și parola cu care vă conectați la serverul MySQL) în fișiere cu extensia .inc, .txt sau .html care pot fi accesate și văzute.

Folosiți pentru acestea fișiere cu extensia .php care, dacă sunt accesate direct, vor fi rulate fără să afișeze informațiile conținute în ele.

Cum spuneam, acestea sunt doar câteva din metodele pe care le puteți folosi pentru a face aplicații sigure. Siguranță absolută nu există însă prevenirea poate fi salvatoare. Mai multe informații privind securitatea puteți găsi la www.secure-reality.com.au și www.securityfocus.com.

...save our values...

**CD, MC, DVD manufacturing
on firm grounds**



Copy protection for CD-Audio and CD-ROM

VTCD VIDEOTON

Compact Disc Factory Ltd.



www.chip.ro www.chip.ro www.chip.ro

www.chip.ro www.chip.ro www.chip.ro

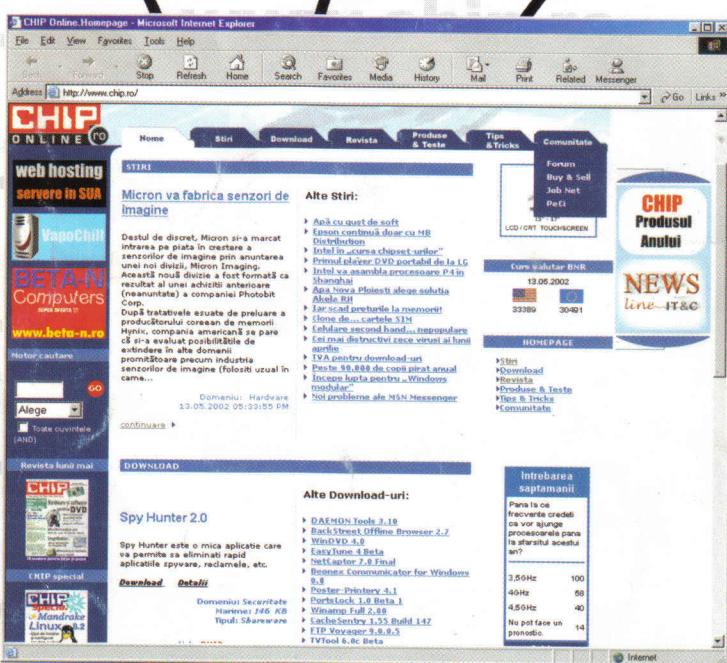
www. CHIP .ro

www.chip.ro www.chip.ro www.chip.ro

Știri

Download

Produse & Teste



**design
attractiv**

**navigare
facilă**

**site-ul tău de legătură cu lumea
tehnologiei informațiilor și comunicației**

www.chip.ro www.chip.ro www.chip.ro

www. CHIP .ro

www.chip.ro www.chip.ro www.chip.ro