Programarea interfețelor web utilizând PHP şi MySQL
Curs 1

Introducere în *PHP* şi MySQL

Obiectivul acestui curs este de a învăța cum să creem site-uri web dinamice, în concordanță cu cerințele actuale. Site-urile web pot fi dezvoltate utilizând *HTML* simplu, însă această abordare are foarte multe limitări, deoarece conținutul unui astfel de site este static, nemodificându-se decât dacă îl actualizăm fizic; utilizatorii nu pot interacționa cu site-ul în mod semnificativ.

Utilizarea unui limbaj precum *PHP* şi a unei baze de date (*MySQL*) permite obținerea unor site-uri web dinamice, care pot fi personalizate şi conțin informație în timp real.

Ce este *PHP*?

PHP este un limbaj de tip *scripting* pe partea de server, proiectat special pentru Web. Într-o pagină *HTML* poate fi încapsulat cod *PHP* care va fi executat de fiecare dată când pagina este vizitată. Codul *PHP* este interpretat pe serverul web şi generează cod *HTML* (sau un alt output) care va fi văzut de către vizitator.

PHP a fost proiectat în 1994 şi, la origine, a fost contribuția unui singur om, Rasmus Lendorf. Apoi, limbajul a fost adoptat şi de către alți specialişti şi a trecut prin 4 rescrieri majore care l-au adus la versiunea complexă de astăzi. În noiembrie 2007, era instalat pe mai mult de 21 milioane de domenii din lumea întreagă, număr aflat în continuă creştere.

PHP este un proiect *open source*, ceea ce înseamnă că avem acces la codul sursă şi îl putem utiliza, modifica şi redistribui gratuit.

La început, acronimul *PHP* reprezenta *Personal Home Page*, dar a fost modificat conform convenției recursive de numire *GNU* (Gnu's Not Unix) şi acum reprezintă *PHP Hypertext Processor*.

Versiunea 5 este versiunea curentă majoră a limbajului *PHP*.

Ce este *MySQL*?

MySQL este un *SGBD* relațional recunoscut pentru rapiditate şi robustețe. Serverul *MySQL* controlează accesul concurent la date de către mai mulți utilizatori, care trebuie să fie autorizați pentru a obține accesul respectiv. Prin urmare, serverul *MySQL* este *multiuser* şi *multi threaded*. El utilizează *SQL*, limbajul standard de gestiune a bazelor de date.

MySQL este disponibil din 1996, însă dezvoltarea sa are originile în anul 1979. Este cel mai popular *SGBD open source*.

MySQL este disponibil sub o schemă de licențiere dublă:

- se poate utiliza sub licență *open source* (*GPL*) în mod gratuit dacă sunt acceptați termenii licenței respective;
- dacă se doreşte distribuirea unei aplicații non-*GPL* care include *MySQL*, poate fi achiziționată o licență comercială.

De ce se utilizează *PHP* şi *MySQL*?

Pentru crearea unui site web pot fi utilizate mai multe produse. În privința deciziilor care trebuie luate, trebuie ales:

- hardware pentru serverul web
- sistemul de operare
- software pentru serverul web
- un *SGBD*
- un limbaj de programare sau de *scripting*.

Unele dintre aceste decizii sunt dependente de celelalte (nu toate sistemele de operare rulează pe orice tip de hardware, nu toate serverele web suportă toate limbajele de programare etc.).

Una dintre trăsăturile importante atât ale *PHP* cât şi *MySQL* este aceea că rulează pe orice sistem de operare important şi multe dintre cele secundare.

Majoritatea codului *PHP* poate fi scris astfel încât să fie portabil între sistemele de operare şi serverele web. Există câteva funcții *PHP* care se referă la sistemul de fişiere şi sunt dependente de sistemul de operare.

Caracteristici ale *PHP*

Principalii concurenți ai *PHP* sunt *Perl*, *Microsoft ASP.NET*, *Ruby*, *Java Server Pages* (*JSP*) şi *ColdFusion*.

Comparativ cu aceste produse, *PHP* are multe avantaje, inclusiv:

- performanță;

- scalabilitate;
- interfețe cu multe *SGBD*-uri;
- biblioteci predefinite pentru multe *task*-uri web comune;
- cost redus;
- facilitatea învățării şi utilizării;
- suport pentru orientarea pe obiecte;
- portabilitate;
- flexibilitate în dezvoltare;
- disponibilitatea codului sursă;
- disponibilitatea suportului şi a documentației.

Performanță. *PHP* este foarte rapid. Utilizând un singur server, necostisitor, putem servi milioane de accesări pe zi.

Scalabilitate. Aşa cum Rasmus Lendorf a spus, *PHP* are o arhitectură „*shared-nothing*". Aceasta presupune că se poate implementa scalarea orizontală cu un număr mare de servere, în mod eficient şi ieftin.

Integrarea bazelor de date. *PHP* are posibilități native de conectare la multe sisteme de baze de date. Pe lângă *MySQL*, se poate conecta direct la *PostgreSQL*, *Oracle*, *DB2*, *Informix*, *InterBase*, *Sybase* etc. Utilizând standardul *ODBC* (*Open Database Connectivity*) ne putem conecta la orice bază de date care furnizează un driver *ODBC*. Aceasta include, printre altele, produsele *Microsoft*.

Pe lângă bibliotecile native, *PHP* vine cu un nivel de abstractizare a accesului la baze de date denumit *PHP Database Objects* (*PDO*), care permite accesul consistent şi promovează practici sigure de scriere a codului.

Biblioteci predefinite. Deoarece *PHP* a fost proiectat pentru *Web*, deține multe funcții *built-in* pentru efectuarea multor *task*-uri referitoare la web. Se pot genera imagini, realiza conexiuni la servicii web şi alte servicii în rețea, parsa cod *XML*, trimite *email*-uri, lucra cu *cookie*-uri şi genera documente *PDF*, toate acestea scriind doar câteva linii de cod.

Cost. *PHP* este gratuit, ultima versiune poate fi descărcată de la adresa http://www.php.net.

Uşurința învățării PHP. Sintaxa *PHP* se bazează pe cea a altor limbaje de programare, în principal *C* şi *Perl*.

Suport pentru orientarea pe obiecte. Versiunea 5 a *PHP* are caracteristici OO bine proiectate: moştenire, atribute şi metode *private* şi *protected*, clase abstracte şi metode, interfețe, constructori şi destructori. De asemenea, există şi câteva trăsături mai puțin comune, cum ar fi iteratorii.

Portabilitate. *PHP* este disponibil pentru multe sisteme de operare: sisteme gratuite *Unix-like* (*Linux*, *FreeBSD*), versiuni *Unix* comerciale (*Solaris*, *IRIX*, *OS X*) sau pe diferite versiuni de *Microsoft Windows*. Un cod *PHP* bine scris va rula fără modificări pe un sistem diferit, care rulează *PHP*.

Flexibilitate în dezvoltare. *PHP* permite implementarea *task*-urilor simple cu uşurință, şi în egală măsură se adaptează la implementarea aplicațiilor mari care utilizează un *framework* bazat pe modele de proiectare (*design patterns*) cum ar fi *Model-View-Controller* (MVC).

Codul sursă. Codul sursă al *PHP* poate fi accesat; este permis să realizăm modificări sau adăugări în limbaj.

Disponibilitatea suportului şi a documentației. Compania din spatele motorului *PHP*, *Zend Technologies* (www.zend.com) oferă suport comercial şi produse software înrudite. Documentația şi comunitatea *PHP* constituie resurse consistente şi mature de informații.

Caracteristici ale MySQL

Principalii concurenți ai *MySQL* sunt *PostgreSQL*, *Microsoft SQL Server* şi *Oracle*. Dintre avantajele *MySQL*, enumerăm:

- performanța ridicată;
- costul scăzut;
- uşurința configurării şi învățării;
- portabilitatea;
- disponibilitatea codului sursă;
- disponibilitatea suportului.

Performanță. Referințele dezvoltatorilor arată că MySQL este cu mult mai rapid decât competitorii săi (http://web.mysql.com/whymysql/benchmarks).

Cost scăzut. *MySQL* este disponibil gratuit sub licență *open source* sau la un cost scăzut sub licență comercială. Este necesară această licență dacă dorim

redistribuirea lui *MySQL* ca parte a unei aplicații. Cele mai multe aplicații web nu trebuie distribuite, astfel încât nu este necesară licența.

Uşurința folosirii. Majoritatea bazelor de date moderne folosesc *SQL*, deci adaptarea la cel implementat în *MySQL* nu ar trebui să fie o problemă. De asemenea, *MySQL* este mai simplu de configurat decât produsele similare.

Portabilitate. *MySQL* poate fi folosit pe diferite sisteme *Unix* şi *Microsoft Windows*.

Codul sursă. Similar limbajului *PHP*, codul sursă al *MySQL* poate fi accesat şi modificat.

Disponibilitatea suportului. Spre deosebire de alte produse *open source*, *MySQL AB* (www.mysql.com) oferă suport, training, consultanță şi certificare.

Limbajul PHP

Pentru început, vom prezenta pe scurt sintaxa şi construcțiile de programare ale limbajului. De asemenea, vom introduce primele exemple pentru a arăta modul în care sunt utilizate variabilele, operatorii şi expresiile în PHP.

Crearea unei aplicații exemplu: magazin de componente auto

Una dintre cele mai comune aplicații ale oricărui limbaj de *scripting* pe partea de server este procesarea formularelor *HTML*. Vom implementa un formular de comandă pentru o companie de piese de schimb pentru maşini. Pe lângă comenzile propriu-zise, proprietarul doreşte să determine prețul total al comenzilor şi taxele aferente acestora.

Inițial, formularul de comandă are următorul conținut: http://193.226.51.37/web/lab1/exemplu1.html

Etapele pentru crearea acestei aplicații sunt scrierea formularului *HTML* şi procesarea acestui formular.

1. Codul *HTML* parțial pentru acest formular se găseşte la adresa: http://193.226.51.37/web/lab1/exemplu1.txt

Observații:

- Acțiunea formularului referă scriptul *PHP* care va procesa comanda clientului (*processorder.php*). În general, valoarea atributului *action* este *URL*-ul care va fi încărcat atunci când utilizatorul acționează butonul *Submit*. Datele pe care utilizatorul le-a introdus în acest formular vor fi trimise către *URL*-ul respectiv prin metoda specificată în atributul *method*, care poate fi *get* (adăugare la sfârşitul *URL*-ului) sau *post* (trimitere ca mesaj separat).
- Numele câmpurilor din formular (*tireqty*, *oilqty*, *sparkqty*) vor fi folosite şi în script-ul *PHP*. Utilizați nume cât mai sugestive! Se recomandă adoptarea unor convenții de scriere a codului, care să prevadă un anumit format pentru numele câmpurilor utilizate în site.

2. Pentru procesarea formularului, trebuie creat scriptul menționat în cadrul *atributului action* din *tag*-ul *form*, *processorder.php*. Codul corespunzător primei versiuni a acestui script se află la adresa: http://193.226.51.37/web/lab1/processorder1.txt

Înlocuind extensia .txt cu .php, vom obține următorul rezultat: http://193.226.51.37/web/lab1/processorder1.php

3. Încapsularea *PHP* în *HTML*.

Sub *heading*-ul <h2> din fişier, adăugăm următoarele linii:

```
<?php
   echo '<p>Order processed.</p>';
?>
```

Fişierul devine: http://193.226.51.37/web/lab1/processorder2.txt

Invocarea acestui fişier va conduce la afişarea următorului formular: http://193.226.51.37/web/lab1/processorder2.php

Observație:

- Dacă vizualizăm sursa paginii în browser, vom observa că nu sunt vizibile liniile *PHP* deoarece interpretorul *PHP* a rulat scriptul şi l-a înlocuit cu *output*-ul acestuia. Aceasta înseamnă că, din *PHP*, putem produce cod *HTML* care poate fi vizualizat cu orice *browser*; cu alte cuvinte, *browser*-ul utilizatorului nu trebuie să compileze *PHP*.

Exemplul anterior ilustrează conceptul de *scripting* pe partea de server. Codul *PHP* a fost interpretat şi executat pe serverul web, spre deosebire de *JavaScript* şi alte tehnologii *client-side* interpretate şi executate în cadrul unui *browser* web pe maşina clientului.

În acest moment, codul pe care îl avem în fişier este de următoarele 4 tipuri:

- *HTML*
- *tag*-uri *PHP*
- instrucțiuni *PHP*
- spații libere.

De asemenea, pot fi adăugate comentarii.

Tag*-urile *PHP sunt <?php şi ?>. Acestea anunță serverul web unde începe şi se termină codul *PHP*, iar orice text dintre aceste 2 *tag*-uri este interpretat ca *PHP*. Textul din afara *tag*-urilor *PHP* este tratat ca *HTML* normal.

Există 4 stiluri diferite de *tag*-uri *PHP*. Următoarele fragmente de cod sunt echivalente:

- stilul XML:

```
<?php echo '<p>Order processed.</p>'; ?>
```

Acesta este stilul de tag *PHP* recomandat. Administratorul serverului nu îl poate dezactiva, deci se poate garanta că va fi disponibil pe toate serverele. Acest stil de tag poate fi utilizat în documentele XML.

- stilul prescurtat:

```
<? echo '<p>Order processed.</p>'; ?>
```

Pentru a utiliza acest stil, este nevoie fie să activăm setarea *short_open_tag* în fişierul de configurare, fie să compilăm *PHP* cu *short tag*-urile activate. Stilul nu este activat implicit!

- stilul *script*:

```
<script language='php'>
   echo '<p>Order processed.</p>'; </script>
```

Stilul este familiar celor care au utilizat *JavaScript* sau *VBScript*. Poate fi folosit dacă editorul *HTML* are probleme cu celelalte stiluri.

- stilul *ASP*:

```
<% echo '<p>Order processed.</p>'; %>
```

Poate fi folosit dacă a fost activată setarea de configurare *asp_tags*. Implicit, stilul este dezactivat.

Observații:

- Instrucțiunile PHP se scriu între tag-urile PHP şi sunt separate prin „;”. Un prim exemplu-, apărut mai sus, de instrucțiune PHP este *echo*.
- Spațiile libere (*whitespaces*) includ *newline*, *blank*-uri şi *tab*-uri. *Browser*-ele ignoră aceste spații în *HTML*, iar motorul *PHP* procedează similar. Aceste spații sunt utile pentru a obține un cod sursă mai lizibil.
- Comentariile în *PHP* pot fi în stil *C* (/* */), *C++* (//) sau *shell script* (#).

Adăugarea conținutului dinamic

Motivul principal pentru care utilizăm un limbaj de *scripting* pe partea de server este furnizarea de conținut dinamic pentru utilizatorii unui site.

De exemplu, modificăm codul *PHP* din *processorder1.php* astfel încât să obținem data şi ora la care a fost procesată comanda: http://193.226.51.37/web/lab1/processorder3.txt

Rezultatul este: http://193.226.51.37/web/lab1/processorder3.php

Observații:

- Operatorul de concatenare este „.”.
- Funcția predefinită *date*() acceptă ca argument un şir de caractere ce reprezintă formatul datei. Dintre formatele existente, amintim: *H* este ora în formatul cu 24 de ore, *i* reprezintă numărul de minute cu „0” la început, dacă este necesar, *j* este ziua din lună fără „0” la început, *s* reprezintă sufixul ordinal (*th*), *F* este numele întreg al lunii.

Accesarea variabilelor din formulare

Scopul utilizării formularelor este de a obține informații furnizate de către clienți. Obținerea acestor detalii este simplă în *PHP*, dar metoda exactă depinde de versiunea de *PHP* utilizată şi de o setare din fişierul *php.ini*.

Variabile

În cadrul *script*-ului *PHP*, fiecare câmp poate fi accesat ca o variabilă *PHP* al cărei nume este în legătură cu numele câmpului respectiv. Numele variabilelor sunt uşor de recunoscut, deoarece încep cu semnul „$”.

În funcție de versiunea de *PHP* şi de configurare, datele din formular pot fi accesate în 3 moduri. Pentru a accesa conținutul câmpului *tireqty*, aceste modalități sunt:

- *$tireqty*
- *$_POST['tireqty']*
- *HTTP_POST_VARS['tireqty']*

Pentru a alege una dintre cele 3 modalități, trebuie avute în vedere următoarele considerente:

- Stilul „scurt” (*$tireqty*) este convenabil, însă necesită ca setarea de configurare *register_globals* să fie activată. Din motive de securitate, această setare este implicit dezactivată. Acest stil facilitează introducerea unor erori care determină insecuritatea codului, motiv pentru care nu este o abordare recomandată (pe lângă aceasta, se pare că opțiunea va dispărea într-o versiune viitoare).
- Stilul „mediu” (*$_POST['tireqty ']*) constituie abordarea recomandată. Dacă vom crea versiuni scurte ale numelor variabilelor, pe baza stilului „mediu”, nu mai apare o problemă de securitate.
- Stilul „lung” (*$HTTP_POST_VARS['tireqty']*) oferă cele mai multe informații. Oricum, acest stil este depreciat şi probabil că va fi eliminat ulterior. Acest stil a fost cel mai portabil, însă acum poate fi dezactivat prin intermediul directivei de configurare *register_long_arrays*, care îmbunătățeşte performanța.

Revenind la stilul „mediu”, acesta presupune regăsirea variabilelor din formular într-unul dintre tablourile *$_POST*, *$_GET* sau *$_REQUEST*. Unul dintre tablourile *$_GET* sau *$_POST* reține detalii despre toate variabilele din formular. Tabloul folosit depinde de tipul metodei de trimitere a

informațiilor din formular (*GET* sau *POST*). O combinație a tuturor datelor trimise prin *GET* sau *POST* se află în tabloul *$_REQUEST*.

Aceste tablouri sunt superglobale.

Pentru uşurința folosirii, se pot crea copii ale variabilelor utilizând operatorul de atribuire (=). Următoarea instrucțiune creează o nouă variabilă denumită *$tireqty* în care este copiat conținutul lui *$_POST['tireqty ']*:

```
$tireqty = $_POST['tireqty'];
```

Pentru a trata cu mai multă uşurință datele dintr-un formular, se poate plasa la începutul *script*-ului un fragment de cod de forma următoare:

```
<?php
// crearea variabilelor “scurte”
$tireqty = $_POST['tireqty'];
$oilqty = $_POST['oilqty'];
$sparkqty = $_POST['sparkqty '];
?>
```

Pentru ca script-ul să producă un rezultat vizibil, afişăm valorile variabilelor în script (http://193.226.51.37/web/lab1/processorder4.txt). Rezultatul este vizibil rulând o versiune modificată a primului script (exemplu1.html) care permite selectarea formularului de procesare a comenzii (deocamdată, 1-4): http://193.226.51.37/web/lab1/exemplu2.php (cod sursă disponibil în exemplu2.txt).

Concatenarea şirurilor de caractere

Aşa cum am menționat într-o observație anterioară, operatorul de concatenare este „.”.

Variabilele simple pot fi plasate şi în interiorul şirurilor situate între ghilimele("). De exemplu instrucțiunea:

```
echo "$tireqty tires<br />";
```

este echivalentă cu cea din script-urile anterioare.

Procesul de înlocuire a conținutului unei variabile într-un şir de caractere este cunoscut sub numele de interpolare. Aceasta este o trăsătură a şirurilor de caractere plasate între ghilimele, nu apostrofuri!

Variabile şi literale

Variabilele *simbolizează* date, iar şirurile de caractere *sunt* date. Atunci când datele însele sunt utilizate ca atare în programe, ele se numesc

literale, spre a fi deosebite de variabile. *$tireqty* este o variabilă, iar *' tires
 '* este un literal.

Avem două tipuri de şiruri de caractere: cu ghilimele şi cu apostrofuri. *PHP* încearcă să evalueze şirurile de caractere dintre ghilimele, aşa cum am arătat mai sus. Spre deosebire de acestea, şirurile de caractere aflate între apostrofuri sunt tratate ca literale.

Mai există un al treilea mod de specificare a şirurilor de caractere utilizând sintaxa *heredoc* (<<<), familiară utilizatorilor *Perl*. Aceasta permite specificarea şirurilor lungi de caractere, adăugând un *marker* de final care va fi utilizat pentru a încheia şirul.

Următorul exemplu creează şi afişează un şir de caractere de 3 linii:

```
echo <<<theEnd
line 1
line 2
line 3
theEnd
```

Similar şirurilor între apostrofuri, şi cele *heredoc* pot fi interpolate.

Identificatori

Identificatorii sunt nume de variabile, funcţii şi clase.

Regulile pentru definirea de identificatori valizi sunt următoarele:

- Identificatorii pot avea orice lungime şi pot conţine litere, cifre şi *underscore*.
- Identificatorii nu pot începe cu o cifră.
- În *PHP*, identificatorii sunt *case-sensitive*. Funcţiile fac excepţie de la această regulă, numele lor putând fi scrise oricum.
- O variabilă poate avea acelaşi nume cu o funcţie (nu se recomandă acest lucru, deoarece poate produce confuzie).

Pe lângă variabilele corespunzătoare formularelor *HTML*, pot fi declarate şi utilizate propriile variabile.

O trăsătură a limbajului *PHP* este că nu necesită ca variabilele să fie declarate înainte de a fi utilizate. O variabilă este creată atunci când i se atribuie prima dată o valoare.

Presupunem că dorim să se calculeze numărul de articole comandate şi totalul de plată. Putem crea două variabile care să stocheze aceste valori, pe care le iniţializăm cu 0 la începutul script-ului *PHP*:

```
$totalqty = 0;
$totalamount = 0.00;
```

Tipurile de date ale variabilelor

PHP oferă următoarele tipuri de date de bază:

- *Integer* – numere întregi
- *Float* (double) – numere reale
- *String* – şiruri de caractere
- *Boolean* – valori logice
- *Array* – stochează date multiple
- *Object* – stochează instanțe ale claselor.

De asemenea, sunt disponibile două tipuri speciale: *NULL* şi *resource*.

Variabilele cărora nu le-a fost atribuită o valoare sau care au primit valoarea specifică *NULL* sunt de tip *NULL*.

Anumite funcții predefinite (de exemplu, funcțiile referitoare la baza de date) returnează variabile al căror tip este *resource*. Acestea reprezintă resurse externe(cum ar fi conexiunile la baza de date). O astfel de variabilă nu va fi prelucrată direct; de obicei, acestea sunt returnate de anumite funcții şi trebuie transmise ca parametri altor funcții.

Puterea tipurilor de date

PHP este un limbaj cu tipuri de date dinamice (sau slabe). În majoritatea limbajelor de programare, variabilele pot reține un singur tip de date, iar tipul de date trebuie declarat înainte ca variabila să îl poată folosi. În *PHP*, tipul unei variabile este determinat de variabila atribuită lui.

În atribuirile precedente, *$totalqty* este o variabilă de tip întreg, iar *$totalamount* este o variabilă de tip real. Pe de altă parte, se poate adăuga ulterior următoarea linie în script:

```
$totalamount = 'Hello';
```

Variabila *$totalamount* va avea apoi tipul de date *string*. *PHP* modifică tipul de date al variabilei în funcție de ceea ce reține respectiva variabilă la un moment dat.

Conversia tipurilor de date

Putem considera că o variabilă sau valoare este de tip diferit utilizând conversia tipurilor de date. Această caracteristică lucrează identic modului existent în *C*. Tipul de date temporar este plasat între paranteze în fața variabilei.

De exemplu, putem avea următoarele declarații:

```
$totalqty = 0;
$totalamount = (float)$totalqty;
```

Variabile „variabile" (*variable variables*)

PHP furnizează un nou tip: variabilele variabile. Acestea permit modificarea numelui unei variabile în mod dinamic.

O astfel de variabilă utilizează valoarea unei variabile ca nume pentru o altă variabilă. De exemplu, putem avea atribuirea:

```
$varname = 'tireqty';
```

iar apoi putem utiliza $$*varname* în locul lui $*tireqty*. De exemplu, putem seta valoarea lui $*tireqty* după cum urmează:

```
$$varname = 5;
```

Aceasta este echivalentă cu:

```
$tireqty = 5;
```

În loc de a lista şi utiliza fiecare variabilă din formular separat, se poate utiliza o ciclare şi o variabilă pentru a le procesa pe toate automat.

Constante

O constantă stochează o valoare, asemănător unei variabile, însă această valoare este stabilită o singură dată şi nu poate fi modificată ulterior.

În aplicația exemplu, putem stoca prețurile pentru fiecare item de vânzare sub forma unei constante. Constantele se declară utilizând funcția *define*.

```
define('TIREPRICE', 100);
define('OILPRICE', 10);
define('SPARKPRICE', 4);
```

Scrierea cu majuscule a constantelor este doar o convenție care face mai uşoară deosebirea dintre variabile şi constante.

Există câteva deosebiri importante între variabile şi constante:

- referirea unei constante se face fără semnul „$" în față;
- constantele pot stoca doar date de tip scalar (*boolean*, *integer*, *float*, *string*).

Pe lângă constantele definite de utilizator, *PHP* oferă un număr mare de constante predefinite, care pot fi listate cu ajutorul funcției *phpinfo*();

Domeniul de vizibilitate al variabilelor

Regulile de bază referitoare la domeniile de vizibilitate din PHP sunt:

- Variabilele predefinite superglobale sunt vizibile oriunde în *script*.
- Constantele, odată declarate, sunt întotdeauna vizibile global (atât în funcții, cât şi în afara acestora).
- Variabilele globale declarate într-un *script* sunt vizibile în tot *script*-ul, dar nu şi în interiorul funcțiilor.
- Variabilele din interiorul funcțiilor care sunt declarate globale referă variabilele globale având acelaşi nume.
- Variabilele create în interiorul funcțiilor şi declarate statice sunt invizibile din afara funcției dar îşi păstrează valorile de la o execuție a funcției la alta.
- Variabilele create în interiorul funcțiilor sunt locale respectivei funcții şi încetează să existe atunci când funcția îşi încheie execuția.

Tablourile *$_GET* şi *$_POST*, precum şi alte variabile speciale au propriile lor reguli privind vizibilitatea: Acestea sunt cunoscute ca superglobale sau autoglobale şi sunt vizibile de oriunde.

Lista completă a variabilelor superglobale este următoarea:

- *$GLOBALS* – tablou conținând toate variabilele globale; permite accesarea variabilelor globale din interiorul unei funcții (*$GLOBALS[' myvariable ']*.);
- *$_SERVER* – tablou conținând variabilele de mediu ale serverului;
- *$_GET* – tablou cu variabilele transmise script-ului prin intermediul metodei *GET*;
- *$_POST* – tablou cu variabilele transmise script-ului prin intermediul metodei *POST*;
- *$_COOKIE* – tablou cu variabilele din *cookie*-uri;
- *$_FILES* – tablou cu variabilele referitoare la încărcările de fişiere;
- *$_ENV* – tablou cu variabilele de mediu;
- *$_REQUEST* – tablou cu toate intrările din partea utilizatorului, incluzând *$_GET*, *$_POST*, şi *$_COOKIE*;
- *$_SESSION* – tablou cu variabilele sesiunii.

Operatori

1) Operatori aritmetici: +, -, *, /, % (modulo).
2) Operatori pe şiruri de caractere: .
3) Operatorul de atribuire: =

Observație: Pot fi formate expresii de forma: `$b = 6 + ($a = 5);`

4) Operatori de atribuire compuşi:

Operator	Utilizare	Echivalent cu
+=	$a += $b	$a = $a + $b
-=	$a -= $b	$a = $a - $b
*=	$a *= $b	$a = $a * $b
/=	$a /= $b	$a = $a / $b
%=	$a %= $b	$a = $a % $b
.=	$a .= $b	$a = $a . $b

5) Pre- şi Post- incrementare şi decrementare: ++, - - sunt similari operatorilor += şi -=, cu diferența că atribuirea are loc anterior, respectiv ulterior, operației propriu-zise.

6) Operatorul de referință: &
   Când o variabilă este atribuită unei altei variabile, se face o copie a sa şi se stochează în memorie. Liniile următoare au acest efect (se creează o copie a lui *$*a şi se stochează în *$b*):
```
$a = 5;
$b = $a;
```
   În acest exemplu, o modificare ulterioară a valorii lui *$a* nu are niciun efect asupra valorii lui *$b*. Pentru a nu fi realizată o copie, se foloseşte operatorul referință:
```
$a = 5;
$b = &$a;
$a = 7; // $a si $b au acum valoarea 7
```
   Atât *$a* cât şi *$b* referă aceeaşi locație de memorie. Acest lucru poate fi modificat cu ajutorul funcției *unset*() (de exemplu, *unset*(*$a*)).

7) Operatori de comparare:
   - Operatorul de egalitate: == (valorile stocate în cele 2 variabile sunt egale?)
     Atenție! Confuzia între = şi == poate conduce la următoarea situație, deoarece valorile diferite de 0 sunt evaluate *true*, iar 0 este evaluat *false*:
```
$a = 5;
$b = 7;
$a = $b // true; valoarea este cea din membrul stang, adica 7 (=> true)
```
   - Operatorul de identitate: ===returnează *true* doar dacă operanzii sunt egali şi au acelaşi tip de date (0=='0' true, dar 0 === '0' false).

- Operatorii !=, !==, <>, <, >, <=, >=

8) Operatori logici: AND (&&, *and*), OR (||, *or*), XOR (*x or*), NOT (!).
Observație: Operatorii and şi or au precedență mai mică decât && şi ||.

9) Operatori pe biți: AND (&), OR (|), NOT (~), XOR (^), left shift (<<), right shift (>>).

10) Operatorul ternar:
```
condition ? value_if_true : value_if_false
```

11) Operatorul de suprimare a erorilor: @
Exemplu: `$a = @(57/0);`

12) Operatorul de execuție: `` - ceea ce este plasat între apostrofurile inverse este executat ca o comandă în linia de comandă a serverului.
Exemplu: (în Windows)
```
$out = `dir c:`;
echo '<pre>'.$out.'</pre>';
```

13) Operatori pe tablouri: reuniunea (+), egalitatea (==), identitatea (===), inegalitatea (!=, <>, non-identitatea (!==).

14) Operatorii pentru instanțiere: *new*, ->

15) Operatorul pentru tipuri: *instanceof* (utilizat în programarea orientată pe obiecte) – permite să aflăm dacă un obiect este o instanță a unei anumite clase.

Efectuarea totalurilor în formular

Pentru a calcula cantitatea totală de articole comandate, precum şi taxele aferente acestora, procesarea din aplicația exemplu se modifică astfel: http://193.226.51.37/web/lab1/processorder5.txt

Scriptul anterior utilizează operatori (+, *, .) şi funcția *number_format*() – pentru a formata totalurile ca şiruri de caractere cu două zecimale.

Precedența şi asociativitatea

În general, operatorii au o anumită precedență sau ordine în care sunt evaluați. De asemenea, ei au şi o asociativitate, care se referă la ordinea în

care sunt evaluați operatorii de aceeaşi precedență. În general, ordinea poate fi de la stânga spre dreapta (*left*), de la dreapta spre stânga (*right*) sau *not relevant*.

Tabelul următor arată precedența şi asociativitatea operatorilor în *PHP*. Operatorii sunt ordonați în ordinea crescătoare a precedenței.

Associativitate	**Operatori**
left	,
left	or
left	xor
left	and
right	print
left	=
+=	-=
*=	/=
.=	%=
&=	|=
^=	~=
<<=	>>=
left	?
:	
left	||
left	&&
left	|
left	^
left	&
n/a	==
!=	===
!==	
n/a	<
<=	>
>=	
left	<<
>>	
left	+
-	.
left	*
/	%
right	!
~	++
--	(int)
(double)	(string)
(array)	(object)
@	
right	[]
n/a	new
n/a	()

Observații:

- Operatorul având cea mai mare precedență este reprezentat de paranteze (). Acestea ridică precedența operațiilor pe care le conțin.
- Operatorul *print* este echivalent cu comanda *echo*. Nici *print*, nici *echo* nu sunt funcții, însă pot fi apelate ca funcții cu parametri în paranteze. De asemenea, amândoi pot fi tratați ca operatori: şirul de caractere se specifică după cuvântul cheie respectiv. Apelul lui *print* ca funcție returnează valoarea 1.

Utilizarea funcțiilor pentru variabile

PHP furnizează o bibliotecă de funcții care permit prelucrarea şi testarea variabilelor în diferite moduri.

Testarea şi setarea tipurilor de variabile

Cele mai multe funcții pentru variabile se referă la testarea tipului unei variabile. Funcțiile cele mai generale sunt *gettype*() şi *settype*(), ale căror prototipuri sunt:

```
string gettype(mixed var);
bool settype(mixed var, string type);
```

Observație: Documentația *php.net* face referință la tipul de date „*mixed*”. Acest tip nu există, dar pentru că *PHP* este atât de flexibil în privința tipurilor, multe funcții pot accepta mai multe tipuri de date (sau orice tip de dată) pentru argumente. Astfel de argumente sunt afişate având pseudo-tipul „*mixed*”.

Exemplu:

```
$a = 56;
echo gettype($a).'<br />'; //integer
settype($a, 'double');
echo gettype($a).'<br />';//double
```

PHP furnizează şi câteva funcții specifice pentru testarea tipurilor. Fiecare dintre acestea are ca argument o variabilă şi returnează *true* sau *false*. Aceste funcții sunt:

- *is_array*() – verifică dacă variabila este un tablou;
- *is_double*(), *is_float*(), *is_real*() – verifică dacă variabila este reală;

- *is_long*(), *is_int*(), *is_integer*() – verifică dacă variabila este întreagă;
- *is_string*() – verifică dacă variabila este de tip şir de caractere;
- *is_bool*() – verifică dacă variabila este de tip Boolean;
- *is_object*() – verifică dacă variabila este un obiect;
- *is_resource*() – verifică dacă variabila este o resursă;
- *is_null*() – verifică dacă variabila este *null*;
- *is_scalar*() – verifică dacă variabila este scalară.
- *is_numeric*() – verifică dacă variabila este un număr (întreg sau real) sau un şir de caractere ce reprezintă un număr;
- *is_callable*() – verifică dacă variabila reprezintă numele unei funcții valide.

Testarea stării variabilelor

PHP oferă funcții pentru testarea stării variabilelor.

- `bool isset(mixed var);[;mixed var[,...]])` – funcția primeşte ca argument numele unei variabile şi returnează *true* dacă aceasta există şi *false* altfel. De asemenea, se poate transmite o listă de variabile, caz în care funcția returnează *true* dacă toate aceste variabile sunt definite.
- `void unset(mixed var);[;mixed var[,...]])`– elimină variabila transmisă ca argument.
- `bool empty(mixed var)` – verifică dacă variabila există şi are o valoare nevidă, diferită de 0.

Exemplu: Presupunem că este introdus următorul cod în script-ul considerat drept exemplu. Valorile returnate de funcții sunt scrise în comentariu.

```
echo 'isset($tireqty): '.isset($tireqty).'br />'; // true
echo 'isset($nothere): '.isset($nothere).'<br />'; //false
echo 'empty($tireqty): '.empty($tireqty).'<br />';
                            //depinde de ce a fost introdus
echo 'empty($nothere): '.empty($nothere).'<br />';//true
```

Reinterpretarea variabilelor

Echivalentul conversiei variabilelor se poate obține cu ajutorul unor funcții:

- `int intval(mixed var[, int base]);`
- `float floatval(mixed var);`
- `string strval(mixed var);`

Fiecare dintre aceste funcții primeşte ca argument o variabilă şi returnează valoarea variabilei convertită la tipul respectiv. Funcția *intval* permite specificarea bazei pentru conversie atunci când variabila este convertită la un string. Astfel, se poate converti, de exemplu, un şir hexazecimal în număr întreg.