

PHP & MySQL:

Εισαγωγή στην Ανάπτυξη Δικτυακών Εφαρμογών

Συγγραφείς : Καρατζίδης Στράτος
 Αθανασιάδης Ιωάννης

Διδάσκων : Αθανασιάδης Ιωάννης

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΙΣΑΓΩΓΗ ΣΤΗΝ PHP ΚΑΙ ΣΤΗΝ ΑΝΑΠΤΥΞΗ ΔΙΚΤΥΑΚΩΝ ΕΦΑΡΜΟΓΩΝ.....	4
Γενικά για την PHP.....	4
Server-side scripting.....	4
HTTP protocol.....	6
MIME types.....	7
Περιβάλλον ανάπτυξης	8
Εφαρμογές / Παραδείγματα	8
Basic Language Syntax.....	9
Comments.....	9
Μεταβλητές (Variables).....	9
Γενικά.....	9
Data Types.....	10
Environment Variables – Predefined Variables.....	13
Δυναμική διαχείριση μεταβλητών (Variable Variables).....	14
Constants.....	14
Operators.....	15
Math operators.....	15
String Operators.....	15
Operator precedence.....	15
Comparison operators.....	16
Logical Operators.....	16
References.....	17
Μετατροπές / Έλεγχος τύπου μεταβλητής.....	19
HTML forms processing.....	22
HTML / web forms / Collecting Data.....	22
Τρόποι αποστολής δεδομένων.....	22
Πέρασμα μεταβλητών και δεδομένων με HTML forms.....	22
Tips for HTML, forms.....	23
Control Structures and Loops.....	24
IF, IF /ELSE, ELSEIF, Nested IF.....	24
Switch statement.....	25
For loops.....	27
While loops	28
Do... While.....	28
Break.....	29
Continue.....	29
Files, Directories and Email.....	31
File operations (open, close, read, write, append).....	31
Ownership – Permissions – Directories.....	32
Copy / Upload Files.....	32
Include / require.....	33
Email.....	34
Headers.....	34
Caching.....	34
Άλλοι τρόποι χρησιμοποίησης της header().....	34
Τεχνικές / Πρακτικές για τη συγγραφή προγραμμάτων / scripts.....	34
Functions and Arrays.....	36
Function (purpose, defining, calling).....	36
Pass Arguments to a function (by value – by reference – by default value).....	37
Variable scope (global και local variables μέσα σε functions).....	37
Nested Functions.....	38
Arrays	38
Multidimensional Arrays.....	39

Foreach και while/list/each loops: επαναληπτικές δομές για πίνακες.....	40
Cookies and Sessions.....	43
Describe persistence (γιατί την χρειαζόμαστε), hidden input fields to pass variables.....	43
Cookies.....	43
Create / read a cookie.....	44
read / retrieve a cookie.....	44
Delete a cookie.....	44
Sessions.....	45
Δημιουργία των sessions.....	45
Mysql and PHP.....	47
Access MySQL using command mode / basic SQL syntax.....	48
Εργαλεία διαχείρισης MySQL.....	49
Βασικές εντολές SQL.....	49
Δημιουργία μιας Βάσης Δεδομένων.....	49
Διαγραφή μιας ΒΔ.....	49
Διαχείριση.....	49
Null / Not Null.....	50
Value range.....	51
Select.....	55
Insert.....	59
Update.....	59
Delete.....	59
Ενώσεις (Joins).....	59
Cross/Cartesian Join.....	60
Connecting in a mysql database using PHP.....	60
Basic mysql functions in PHP.....	61
PHP and mysql tips.....	61

ΕΙΣΑΓΩΓΗ ΣΤΗΝ PHP ΚΑΙ ΣΤΗΝ ΑΝΑΠΤΥΞΗ ΔΙΚΤΥΑΚΩΝ ΕΦΑΡΜΟΓΩΝ

Γενικά για την PHP

Ξεκίνησε το 1995 από τον Rasmus Lerdorf.

PHP: Personal Home Pages. Άλλαξε το 1997 σε Hypertext Preprocessor.

Γνωρίσματα και Πλεονεκτήματα της PHP:

- Εξοικείωση (familiarity): η PHP συντακτικά μοιάζει πολύ με άλλες γλώσσες προγραμματισμού (C, Perl), οπότε είναι εύκολη η προσαρμογή και η διαδικασία εκμάθησης.
- Αποτελεσματικότητα: διαθέτει όλα τα features που χρειάζονται για όλες τις απαιτητικές δουλειές στο web (sessions, sockets, ftp κ.λ.π.)
- Ασφάλεια (encryption κ.λ.π.)
- Ευκαμψία. Συνδυάζεται με άλλες γλώσσες (XML, Javascript, HTML, Shell Programming).
- Επίσης εγκαθίσταται σχεδόν σε όλα τα Operating Systems και Web Servers
- Είναι δωρεάν

Επίσης η PHP είναι:

- Cross-Platform. Ο ίδιος PHP κώδικας τρέχει σχεδόν σε όλες τις πλατφόρμες
- Server-Side. Τα PHP scripts εκτελούνται στον web server, όχι στον client / browser.
- HTML Embedded. Το script ενσωματώνεται σε HTML σελίδα ανάμεσα σε <?...?> σύμβολα. Scripting Language. Τα PHP scripts εκτελούνται σε servers, δεν είναι stand-alone προγράμματα.

Server-side scripting

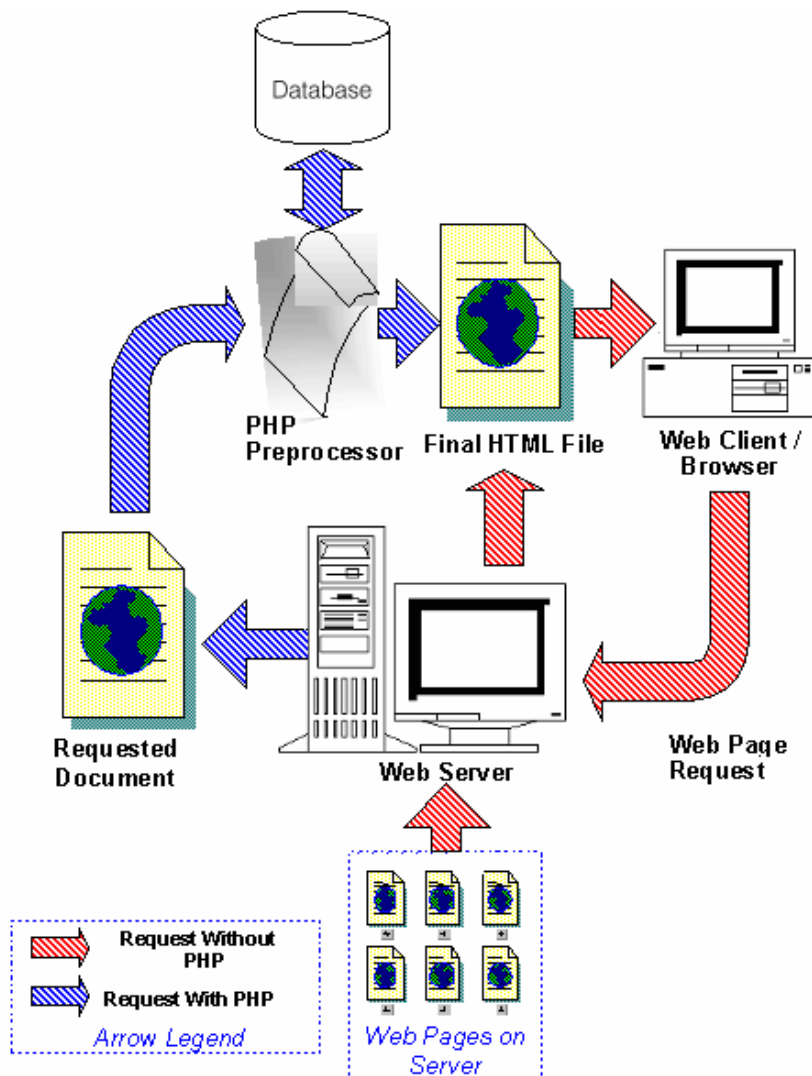
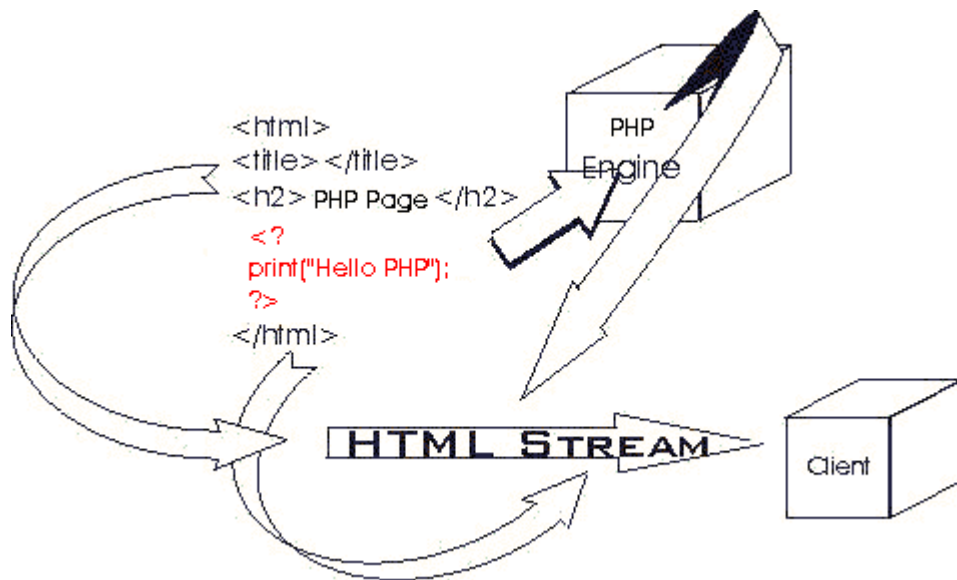
Χρειάζεται μια γλώσσα που να παράγει δυναμικό, ανανεώσιμο περιεχόμενο και όχι απλά στατικές html σελίδες.

Τα scripts εκτελούνται στον server, όχι στον client

Το output είναι απλές, html σελίδες, ενώ αντίθετα η επεξεργασία γίνεται με μια γλώσσα προγραμματισμού (π.χ. Perl, PHP, Java, Visual Basic).

Η html σελίδα στέλνεται από τον server και επεξεργάζεται και παρουσιάζεται από τον browser.

Όταν η PHP μεταγλωττίζει (parses) ένα αρχείο, απλά κάνει ένα πέρασμα στο κείμενο του αρχείου μέχρι να συναντήσει ένα από τα ειδικά tags που της λένε να αρχίσει να μεταφράζει το κείμενο ως κώδικα PHP. Ο parser (μεταγλωττιστής) τότε εκτελεί ολόκληρο τον κώδικα που βρίσκει, μέχρι να συναντήσει το επόμενο PHP tag κλεισίματος, το οποίο λέει στον parser να αρχίσει να κάνει ξανά, απλά ένα πέρασμα στο κείμενο. Αυτός είναι ο μηχανισμός που σας επιτρέπει να προσθέτετε PHP κώδικα μέσα σε HTML: ο,τιδήποτε βρίσκεται έξω από τα tags της PHP μένει τελείως μόνο, ενώ οτιδήποτε μέσα μεταγλωττίζεται ως κώδικας.



HTTP protocol

Χρήση του TCP/IP για τη μεταφορά data / information στο δίκτυο

HTTP: Επιτρέπει στους web browsers να επικοινωνούν με τους web servers και να ανταλλάσσουν πληροφορίες. Σύμφωνα με αυτό το πρωτόκολλο, ο client (browser) στέλνει ένα αίτημα (request) και ο server αποκρίνεται (response). Το μήνυμα που στέλνεται από τον browser (client) στο server λέγεται HTTP request και το μήνυμα από το server στον client λέγεται HTTP response.

Είναι stateless protocol: μετά την απόκριση του server, τερματίζεται η συνδιαλλαγή τους και επομένως δεν είναι διαθέσιμες στη συνέχεια πληροφορίες για τη σύνδεση που πραγματοποιήθηκε

Το μήνυμα αποτελείται από 3 τμήματα: 1.request/response line, 2.HTTP header and 3.HTTP body

REQUEST

1.Request Line: contains the method (GET, HEAD, POST, PUT, DELETE, TRACE, CONNECT, OPTIONS), the path and the HTTP version.

π.χ.

GET /testpage.htm HTTP/1.1

Η γενική μορφή είναι: <METHOD> <request uri> <HTTP version>

GET method: ανάκτηση πληροφοριών από το server, χρησιμοποιείται κυρίως για τη ζήτηση σελίδων / documents από τον server. Επίσης με το GET μπορούμε να στείλουμε πληροφορίες στον server (συνήθως σε CGI πρόγραμμα) σαν τμήμα του URL (με τη χρήση ?).

HEAD method: το ίδιο με το GET, μόνο που δεν επιστρέφει ο server σελίδα, αλλά μόνο το HEADER. Χρήσιμο για περιπτώσεις επιβεβαίωσης ύπαρξης της σελίδας, ελέγχου του file type και της ημερομηνίας αλλαγής της σελίδας κ.λ.π.

POSt method: επιτρέπει στο server να λαμβάνει πληροφορίες από τον client. Τα δεδομένα συνήθως περνάνε με τη μορφή φόρμας στο BODY τμήμα της HTTP request.

2.HTTP headers: Γενικές πληροφορίες για τη σύνδεση.Κάθε header έχει ένα όνομα και μια τιμή και ορίζεται ένας header ανά γραμμή.

Headers for client: Accept, Cookie, referer, User-Agent κ.α.

ACCEPT: */*

ACCEPT_LANGUAGE:en-us

REFERER:http://www.simplygraphix.com/wedes.html

USER_AGENT:Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)

...

Χρησιμοποιούμε μια κενή γραμμή για να ορίσουμε το τέλος του HTTP header.

3. HTTP body

Είναι κενό (όταν ζητάμε μια σελίδα) ή μπορεί να περιέχει τα δεδομένα που στέλνει ο browser με μια φόρμα.

π.χ.

GET /testpage.htm HTTP/1.1

ACCEPT: */*

ACCEPT_LANGUAGE:en-us

REFERER:http://www.simplygraphix.com/wedes.html

USER_AGENT:Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)

Status
Line
Headers

Blank
Line
Body

RESPONSE

1.Response: είναι το status line

περιέχει 1.την έκδοση πρωτοκόλλου του HTTP, 2. ένα κωδικό απόκρισης (response status code) και 3. μια περιγραφή του κωδικού απόκρισης

π.χ.

HTTP/1.0 200 OK ή HTTP/1.0 404 Not Found

2.HTTP headers: Γενικές πληροφορίες για τη σύνδεση.Κάθε header έχει ένα όνομα και μια τιμή και ορίζεται ένας header ανά γραμμή.

Headers for server: Server, Set-Cookie, Last-Modified κ.α.

Χρησιμοποιούμε μια κενή γραμμή για να ορίσουμε το τέλος του HTTP header.

3. HTTP body

Περιέχει το έγγραφο / σελίδα ή δεδομένα που έχει στείλει πίσω ο server

π.χ.

HTTP/1.1. 200 OK

Date: Mon, 04 Jan 2003 12:40:10 GMT

Server: Apache/1.3.27 (Unix)

Last-Modified: Tue, 20 Oct 2002 21:00:39 GMT

Content-Length: 49

Content-Type: text/html

Status Line
Headers

Blank Line
Body

<HTML>

...

</HTML>

MIME types

Multipurpose Internet Mail Extensions: ένα σύνολο από κανόνες που επιτρέπει σε multimedia documents να εναλλάσσονται ανάμεσα σε διαφορετικές πλατφόρμες. Αρχικά χρησιμοποιήθηκε για την επισύναψη αρχείων στο email. Ο web server πρέπει να καθορίσει τον τύπο αρχείου (MIME type) πριν το αποστείλει στον browser και για το σκοπό αυτό εξετάζει το extension του αρχείου, για να βρεί από μια database (συνήθως ένα text αρχείο με όνομα MIME.types) το κατάλληλο MIME type. Υπάρχουν 7 κατηγορίες και η κάθε μια διαθέτει υποκατηγορίες: application, audio, image, message, multipart, text, video. Π.χ. μια HTML σελίδα είναι text Και η υποκατηγορία της είναι html, οπότε το MIME της είναι: text/html, ενώ μια GIF εικόνα έχει MIME type: image/gif.

Περιβάλλον ανάπτυξης

Επίσημο site: www.php.net

Editors: υπάρχουν πάρα πολλοί διαθέσιμοι, ανάλογα με την πλατφόρμα και την άδεια χρήσης
- <http://www.thelinuxconsultancy.co.uk/phpeditors/>

Παραμετροποίηση της μηχανής PHP: αρχείο php.ini

Η PHP μπορεί να λειτουργεί ως:

- shell γλώσσα στο Λειτουργικό Σύστημα. Μπορείτε να φτιάξετε ένα PHP script για να το τρέχετε χωρίς server ή browser. Χρειάζεστε μόνο τον PHP μεταγλωττιστή για να την χρησιμοποιήσετε με αυτό τον τρόπο. Αυτός ο τύπος είναι ιδανικός για script που εκτελούνται συχνά με τη χρήση της cron (σε *nix ή Linux) ή με τον Task Scheduler (στα Windows). Αυτά τα script μπορούν επίσης να χρησιμοποιηθούν για απλές εργασίες επεξεργασίας κειμένου.
- module του apache
- CGI.
- Παραθυρικές εφαρμογές με τη χρήση του PHP-GTK (<http://gtk.php.net/>)

LAMP (Linux, Apache, Mysql, PHP). Εκτελείται σχεδόν σε όλα τα γνωστά Operating Systems.

Εφαρμογές / Παραδείγματα

CMS (www.uom.gr, <http://phpwebsite.appstate.edu/>)

Groupware / Calendar (www.phpprojekt.com)

Forums (www.phpbb.com)

E-commerce (www.oscommerce.com)

Web E-mail systems (<http://www.horde.org/imp/>)

Basic Language Syntax

Ο PHP κώδικας γράφεται μέσα σε ειδικές αγκύλες:

- <?php ... ?>
- <? ... ?>
- <script language="php"> ... </script>
- ASP style: <% ... %>

Οι εντολές διαχωρίζονται με τον ίδιο τρόπο όπως και στην C ή την Perl - τερματίζουμε κάθε εντολή με ένα ερωτηματικό (;)

Το tag κλεισίματος (?>) επίσης υποδηλώνει το τέλος μιας έκφρασης-δηλώσης, συνεπώς τα ακόλουθα είναι ισοδύναμα:

```
<?php
    echo "This is a test";
?>

<?php echo "This is a test" ?>
```

Comments

Η PHP χρησιμοποιεί το στυλ της C στα σχόλια, καθώς και τα σχόλια που χρησιμοποιούμε στο Shell

```
<html>
<title>PHP Comments</title>
<?
echo "Three Kinds of PHP comments";
//This is a PHP single line comment;
/*This is a
multiple line
comment*/
echo "This is a test";
echo "One Final Test"; # This is shell-style style comment
?>
<!--This of course is an HTML style comment-->
</html>
```

1^{ος} τρόπος: // ή # (για κάθε γραμμή)

2^{ος} τρόπος: /* */ (για block γραμμών)

3^{ος} τρόπος: # comments

Είναι καλή πρακτική, να σχολιάζεται πάντα ο κώδικας για μελλοντική επεξεργασία / αλλαγή.

Μεταβλητές (Variables)

Μια περιοχή της μνήμης που ορίζεται από τον προγραμματιστή για να αποθηκεύει προσωρινά δεδομένα.

Γενικά

Οι μεταβλητές ξεκινούν με \$.

Αντιστοιχίζεται η τιμή με το =

Όλες οι μεταβλητές πρέπει να ξεκινούν με letter ή _ (underscore).

Δεν επιτρέπεται μια μεταβλητή να έχει τα +, -, &, *

Το όνομα της μεταβλητής απαγορεύεται να ξεκινάει με ψηφίο

Παραδείγματα μεταβλητών:

```
$today="Monday"; //(κείμενο, string)
```

```
$total=1000; //(number)
```

Οι μεταβλητές είναι case-sensitive

π.χ. \$total και \$Total είναι διαφορετικές μεταβλητές.

Ωστόσο μπορεί να αλλάζει η τιμή της μεταβλητής συνέχεια:

```
$total=10;  
$total=100;
```

Είναι καλή προγραμματιστική τεχνική, πάντα να αρχικοποιούνται οι μεταβλητές πριν την χρήση τους. Ωστόσο στην php δεν είναι απαραίτητο να οριστεί πρώτα η μεταβλητή και μετά να της αποδοθεί τιμή. Μπορούμε κατευθείαν αν της αποδώσουμε τιμή και κατ' αυτόν τον τρόπο έχει οριστεί ως μεταβλητή.

Data Types

Μπορεί να είναι:

- string.

Μια ακολουθία από χαρακτήρες

Το string ορίζεται μέσα σε διπλά ή μονά εισαγωγικά.

Υπάρχει ωστόσο διαφορά στο χειρισμό ανάμεσα σε διπλά ή μονά εισαγωγικά (the difference between " and ')

This will print "This is a text string."

```
<?  
$string = "is a";  
$text = "This $string text string."  
echo $text;  
?>
```

This will print "This \$string text string."

```
<?  
$string = "is a";  
$text = 'This $string text string.'  
echo $text;  
?>
```

The following prints "Some text then PHP"

```
<?  
$a = "PHP";  
echo $b = "Some text then $a";  
?>
```

The following prints "There are 31 days in this month"

```
<?  
$str = "There";  
$str = $str. " are ";  
$str .= date("t");  
$str .= " days in this month."  
echo $str;  
?>
```

The following prints "The value or the variable \$var is 5."

```
<?  
$var = 5;  
$str = 'The value or the variable $var is '$var.'';  
echo $str;  
?>
```

Μεταβλητές τύπου string μέσα σε double quotes επεξεργάζονται (parsed) οι τιμές τους και γίνονται τμήμα του string, ενώ αυτές που είναι σε single quotes όχι.

```
$myVar = "xyz"; // assign a value to $myVar  
echo $myVar; // writes 'xyz'  
echo "abc to $myVar"; // writes 'abc to xyz'  
// but with single quotes  
echo 'abc to $myVar'; // writes 'abc to $myVar'
```

Άλλος τρόπος χειρισμού των strings: heredoc syntax ("<<<"). Δίνουμε ένα αναγνωριστικό (identifier) μετά το <<<, στη συνέχεια το string, και τέλος το ίδιο αναγνωριστικό (identifier) για να τερματίσει. Το identifier που κλείνει το string πρέπει να βρίσκεται στην

αρχή της γραμμής. Το Heredoc text συμπεριφέρεται όπως ένα double-quoted string, χωρίς τα διπλά εισαγωγικά (δεν χρειάζεται δηλαδή να αποφεύγονται τα εισαγωγικά, ενώ μπορούν μέσα να χρησιμοποιούνται και μεταβλητές). Είναι χρήσιμο για περιπτώσεις μεγάλων strings.

```
<?php
$str = <<<EOD
Example of string
spanning multiple lines
using heredoc syntax.
EOD;
?>
```

- numeric

- integer. Integer είναι ένας ολόκληρος αριθμός (χωρίς δεκαδικό τμήμα). Είναι ένας αριθμός ανάμεσα σε -2,147,483,648 και +2,147,483,647. Οι ακέραιοι μπορούν να καθοριστούν στο δεκαδικό (με βάση το 10), δεκαεξαδικό (με βάση το 16) ή οκταδικό (με βάση το 8) σύστημα, και προαιρετικά μπορεί να προστεθεί το πρόσημο (- ή +). Αν χρησιμοποιείτε το οκταδικό σύστημα, πρέπει να γράφετε πριν το αριθμό το 0 (μηδέν), και αν χρησιμοποιείτε το δεκαεξαδικό πρέπει να γράφετε πριν τον αριθμό το 0x. Έαν δεν υπάρχει πρόσημο, θεωρείται θετικός αριθμός.

```
$a = 1234; # decimal number
$a = -123; # a negative number
$a = 0123; # octal number (equivalent to 83 decimal)
```

Ένας οκταδικός (*octal number*) είναι κάποιος αριθμός με βάση το 8. Κάθε ψηφίο έχει τιμή ανάμεσα στο 0 και το 7. Οι οκταδικοί χρησιμοποιούνται στους υπολογισμούς γιατί ένας 3-ψήφιος δυαδικός (binary) αριθμός μπορεί να αναπαρασταθεί με ένα μονοψήφιο οκταδικό αριθμό.

```
$a = 0x1A; # hexadecimal number (equivalent to 26 decimal).
```

Ένας δεκαεξαδικός (*hexadecimal number*) είναι κάποιος αριθμός με βάση το 16. Κάθε ψηφίο έχει τιμή από το 0 έως το F. Εφόσον έχουμε μόνο 10 αριθμούς (0-9), χρησιμοποιούμε τα γράμματα A έως F για τις υπόλοιπες δεκαεξαδικές τιμές. Χρησιμοποιούμε τις δεκαεξαδικές τιμές, γιατί κάθε ψηφίο αναπαριστά 4 δυαδικούς (binary) αριθμούς που είναι 4 bits. 8 bits (ή 2 δεκαεξαδικοί) είναι 1 byte.

Αν προσδιορίσετε έναν αριθμό εκτός των ορίων του τύπου integer, θα διερμηνευτεί ως float (κινητής υποδιαστολής). Επίσης, αν κάνετε μια πράξη το αποτέλεσμα της οποίας δίνει αριθμό πέρα από τα όρια του τύπου integer, θα επιστραφεί αριθμός τύπου float (κινητής υποδιαστολής).

```
$large_number = 2147483648;
var_dump($large_number);
// output: float(2147483648)
// this goes also for hexadecimal specified integers:
var_dump( 0x80000000 );
// output: float(2147483648)
```

- Κινητής υποδιαστολής - floating point (float, double or real)

Αριθμοί κινητής υποδιαστολής (συμβολίζονται και ως "floats", "doubles" or "real numbers") μπορούν να προσδιοριστούν χρησιμοποιώντας οποιονδήποτε από τους ακόλουθους τρόπους σύνταξης

```
$a = 1.234;
$b = 1.2e3;
$c = 7E-10;
```

Μερικές φορές αναφέρονται και ως *real* αριθμοί. Αριθμοί με δεκαδικό τμήμα. Είναι κάτι αντίστοιχο με τον double data type στη C.

- array

Ο πίνακας (*array*) είναι μια μεταβλητή που αποθηκεύει ένα σύνολο από τιμές. Η χρήση τους γίνεται με την αναφορά σε ένα δείκτη (*index*) που έχουν μέσα στον πίνακα. Η θέση / ένδειξη τους είναι είτε αριθμητική (indexed), είτε με όνομα (associative).

- object

Η PHP έχει τη δυνατότητα για object-oriented προγραμματισμό (OOP). Το αντικείμενο (object) είναι ένα data type που επιτρέπει όχι μόνο την αποθήκευση δεδομένων (data), αλλά και συναρτήσεων (τρόπους για το πώς θα διαχειριστούμε τα δεδομένα). Τα δεδομένα που αποθηκεύονται, ονομάζονται *properties* ή *attributes* του object. Ο κώδικας / συναρτήσεις του object που επεξεργάζονται τα δεδομένα, ονομάζονται methods of the object.

Ακολουθούμε 2 βήματα στη δημιουργία (construction) ενός object: Πρώτον, δηλώνουμε / ορίζουμε την class ενός object. Η class ορίζει τη δομή του object που θα δημιουργηθεί και ουσιαστικά λειτουργεί ως ένα template, ως ένα πρότυπο. Δεύτερον, ορίζουμε μια μεταβλητή που να είναι της συγκεκριμένης class και αντιστοιχίζουμε στη μεταβλητή τις κατάλληλες τιμές (*instantiate the object*).

1.ορισμός της class

```
class foo
{
    function do_foo()
    {
        echo "Doing foo.";
    }
}
```

2. Δημιουργία του object

```
$bar = new foo;
$bar->do_foo();
```

- unknown type / NULL

Η NULL αντιστοιχίζεται σε μια μεταβλητή που έχει άγνωστη τιμή. Η NULL είναι η μόνη πιθανή τιμή για τον τύπο NULL

Μια μεταβλητή θεωρείται NULL όταν: i)της έχει αντιστοιχηθεί η σταθερά NULL ή ii)δεν της έχει οριστεί ακόμη κάποια τιμή ή iii)έχει μηδενιστεί με την συνάρτηση unset().

- Boolean (TRUE, FALSE)

Όταν μια μεταβλητή ορίζεται ή μετατρέπεται σε boolean, οι ακόλουθες τιμές θεωρούνται FALSE:

-to FALSE

-ο integer 0 (zero) . Επίσης το -1 θεωρείται TRUE, όπως και οποιοδήποτε άλλο μη-μηδενικός (είτε θετικός είτε αρνητικός) αριθμός

-ο float 0.0 (zero)

-το κενό string και το string "0"

-ένας πίνακας με 0 στοιχεία

-ένα [object](#) με καθόλου member variables

-ο ειδικός τύπος NULL

Ένας boolean εκφράζει μια αληθινή τιμή. Μπορεί να είναι είτε TRUE είτε FALSE

Επίσης είναι case-insensitive:

```
<?php
$foo = True; // assign the value TRUE to $foo
?>
```

Ο καθορισμός γίνεται αυτόματα από την php, αποδίδοντας την κατάλληλη τιμή
\$car_type="Toyota"

Η προσθήκη του κενού διαστήματος μπορεί να γίνει είτε με μια επιπλέον μεταβλητή (π.χ. \$space=" " οπότε γίνεται \$car=\$car_type . \$space . \$car_model), είτε γράφοντας το απευθείας
\$car=\$car_type . " " . \$car_model

Επίσης είναι διαφορετικές οι 2 παρακάτω εντολές:

```
echo 2 . 2; // παράγει 22
echo 2.2 //παράγει 2.2
```

Ακόμη είναι διαφορετικού τύπου οι μεταβλητές στις παρακάτω εντολές:

```
$is_number=1;
$is_string="1";
```

Type juggling: Η αλλαγή των τιμών σε μια μεταβλητή και η ταυτόχρονη αλλαγή τύπου μεταβλητής: εάν π.χ. αντιστοιχίσουμε μια string τιμή σε μια μεταβλητή *\$var*, η *\$var* γίνεται τύπου string. Εάν στη συνέχεια αντιστοιχίσουμε μια integer τιμή στη *\$var*, αυτή μετατρέπεται σε integer.

```
$foo = "0"; // $foo is string (ASCII 48)
$foo += 2; // $foo is now an integer (2)
$foo = $foo + 1.3; // $foo is now a float (3.3)
$foo = 5 + "10 Little Piggies"; // $foo is integer (15)
$foo = 5 + "10 Small Pigs"; // $foo is integer (15)
```

Αλλαγή τύπου μεταβλητής: **type casting**

```
$foo = 10; // $foo is an integer
$str = "$foo"; // $str is a string
$fst = (string) $foo; // $fst is also a string
// This prints out that "they are the same"
if ($fst === $str)
{
    echo "they are the same";
}
```

Environment Variables – Predefined Variables

Δίνουν πληροφορίες για HTTP, clients, OS, web server κλπ. Μπορείτε να τις δείτε με την συνάρτηση `phpinfo()`. Π.χ. ο τύπος του browser: `echo $HTTP_USER_AGENT;`
Ουσιαστικά, αυτές οι μεταβλητές συμπεριφέρονται σαν σταθερές (constants). Ωστόσο δεν μπορεί να αλλαχθεί η τιμή τους, ενώ αντίθετα στις δικές μας σταθερές μπορεί να αλλαχθεί.

Επίσης μπορεί να εμφανιστεί η λίστα τους με τον κώδικα:

```
while(list($index,$value) = each($GLOBALS))
{
    echo "<br>" . $index . "=>" . $value;
}
```

`$_SERVER`

Η `$_SERVER` είναι ένας πίνακας που περιέχει πληροφορίες όπως headers, paths, script locations. Δημιουργείται αυτόματα από τον webserver.

`$_ENV`

Δημιουργούνται από το σύστημα στο οποίο τρέχει η PHP

`$_COOKIE`

Οι μεταβλητές που προέρχονται από τη χρήση HTTP cookies. Είναι global στη χρήση τους.

`$_GET`

Οι μεταβλητές που προέρχονται από τη χρήση HTTP GET. Είναι global στη χρήση τους.

`$_POST`

Οι μεταβλητές που προέρχονται από τη χρήση HTTP POST. Είναι global στη χρήση τους.

`$_FILES`

Για αρχεία που γίνονται upload με την μέθοδο HTTP POST. Είναι global στη χρήση τους.

`$_REQUEST`

Τα περιεχόμενα των `$_GET`, `$_POST` και `$_COOKIE`.

`$_SESSION`

Περιέχει τις session variables που είναι διαθέσιμες για το πρόγραμμα που τρέχει.

\$GLOBALS

Περιέχει όλες τις μεταβλητές που έχουν καθολική χρήση στο πρόγραμμα (global scope). Τα ονόματα των μεταβλητών είναι τα κλειδιά / δείκτες του πίνακα.

\$php_errormsg

Η \$php_errormsg είναι μεταβλητή που περιέχει το τελευταίο μήνυμα λάθους

Ισχύει για \$_SERVER, \$_ENV, \$COOKIE, \$_GET, \$_POST, \$_FILES, \$_REQUEST, \$_SESSION, \$GLOBALS : Είναι 'superglobal' ή automatic global, μεταβλητές, δηλαδή είναι διαθέσιμες παντού σε όλο το script. Δεν χρειάζεται να δηλώσουμε global \$_COOKIE ή \$_SERVER για να τις χρησιμοποιήσουμε μέσα σε functions or methods, όπως συμβαίνει με τις \$HTTP_COOKIE_VARS ή \$HTTP_SERVER_VARS. Η \$HTTP_COOKIE_VARS ή \$HTTP_SERVER_VARS περιέχει τις ίδιες πληροφορίες, αλλά δεν είναι αυτόματα συνολικής χρήσης - autoglobal. Επίσης η \$HTTP_COOKIE_VARS και \$_COOKIE ή η \$HTTP_SERVER_VARS και \$_SERVER είναι διαφορετικές μεταβλητές και κατ' αυτόν τον τρόπο τις μεταχειρίζεται η PHP.

Δυναμική διαχείριση μεταβλητών (Variable Variables)

Έστω \$a = "hello";

Μπορούμε να αποδώσουμε ως όνομα μιας μεταβλητής τη τιμή (value) μιας άλλης:

\$a = "world";

Έτσι έχουμε ορίσει 2 μεταβλητές: \$a με τιμή "hello" και \$hello με περιεχόμενα "world".

echo "\$a \${\$a}"; παράγει το ίδιο με την echo "\$a \$hello";

Constants

Χρησιμοποιούνται πάντα σε περιπτώσεις που δεν αλλάζει εύκολα ή συχνά η τιμή μιας μεταβλητής, οπότε ορίζεται ως σταθερά (constant). Όλες οι μεταβλητές εξ' ορισμού (by default) ορίζονται με κεφαλαία γράμματα (upper case), χρησιμοποιώντας τη δεσμευμένη λέξη define:

```
define("NUMBER_OF_MONTHS",12);
```

Για να εμφανίσουμε την σταθερά δεν χρησιμοποιούμε τα εισαγωγικά:

```
echo "number of months: " . NUMBER_OF_MONTHS;
```

Επίσης η php έχει τις δικές της σταθερές (ουσιαστικά μεταβλητές που συμπεριφέρονται ως σταθερές):

```
echo $PHP_OS; //το O.S. που τρέχει η php
```

```
echo $PHP_SELF; // το όνομα του αρχείου / script.
```

Operators

Κάθε λογική πράξη / έκφραση αποτελείται από τους operands και τους operators:

\$sum = \$x1 + \$x2; (\$sum,\$x1,\$x2 είναι οι operands και + ο operator)

Οι operators εκτελούν μια συγκεκριμένη πράξη

Math operators

+ (πρόσθεση)

- (αφαίρεση)

* (πολλαπλασιασμός)

/ (διαίρεση)

% (υπόλοιπο διαίρεσης)

Προσθέτοντας μια μεταβλητή στον εαυτό της:

\$total=\$total+\$new_value //προσθέτει στην παλιά τιμή της total την new_value

Επίσης είναι το ίδιο με το να δηλώσουμε:

\$total += \$new_value;

Επίσης όταν θέλουμε να αυξήσουμε την τιμή μιας μεταβλητής κατά 1, ορίζεται ως:

\$total=\$total+1;

ή \$total += 1;

ή \$total++;

ή ++\$total;

Ωστόσο υπάρχει μια διαφορά ανάμεσα στο ++\$total και στο \$total++ :

Η ++\$total, πρώτα αυξάνει την τιμή της και στη συνέχεια την αποδίδει, ενώ η \$total++, πρώτα την αποδίδει και στη συνέχεια την αυξάνει.

```
$total = 5;
```

```
echo "Should be 5: " . $total++ . "<br />\n";
```

```
echo "Should be 6: " . $total . "<br />\n";
```

```
$total = 5;
```

```
echo "Should be 6: " . ++$total . "<br />\n";
```

```
echo "Should be 6: " . $total . "<br />\n";
```

Operator	Use	Equivalent to
+=	\$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

String Operators

Η τελεία ('.') αποτελεί τον τρόπο ένωσης αλφαριθμητικών:

```
$car=$car_type . $car_model.
```

Επίσης με την ταυτόχρονη ένωση και απόδοση ('.='). Με αυτόν τον τρόπο ενώνει το όρισμα στα δεξιά με το όρισμα στα αριστερά.

```
$a = "Hello ";
```

```
$b = $a . "World!"; // now $b contains "Hello World!"
```

```
$a = "Hello ";
```

```
$a .= "World!"; // now $a contains "Hello World!"
```

Operator precedence

Ορίζει την προτεραιότητα ανάμεσα στους τελεστές και τις εκφράσεις.

Π.χ. στην έκφραση $1 + 5 * 3$, η απάντηση είναι 16 και όχι 18 γιατί ο πολλαπλασιασμός έχει μεγαλύτερη προτεραιότητα από την πρόσθεση. Γενικά ακολουθούνται οι κανόνες προτεραιότητας όπως έχουν οριστεί στα μαθηματικά (αγκύλες, διαίρεση, πολλαπλασιασμός, πρόσθεση, αφαίρεση).

Associativity	Operators
Left	,
Left	or
Left	xor
Left	and
Right	print
Right	= += -= *= /= .= %= &= = ^= <<= >>=
Left	? :
Left	
Left	&&
Left	
Left	^
Left	&
non-associative	==, !=, ===, !==
non-associative	<, <=, >, >=
Left	<<, >>
Left	+, - .
Left	*, /, %
Right	! ~ ++ -- (int) (float) (string) (array) (object) @
Right	[
non-associative	new

Comparison operators

Γίνεται σύγκριση ανάμεσα σε 2 τιμές.

Example	Name	Result
\$a == \$b	Equal	TRUE if \$a is equal to \$b.
\$a === \$b	Identical	TRUE if \$a is equal to \$b, and they are of the same type. (PHP 4 only)
\$a != \$b	Not equal	TRUE if \$a is not equal to \$b.
\$a <> \$b	Not equal	TRUE if \$a is not equal to \$b.
\$a !== \$b	Not identical	TRUE if \$a is not equal to \$b, or they are not of the same type. (PHP 4 only)
\$a < \$b	Less than	TRUE if \$a is strictly less than \$b.
\$a > \$b	Greater than	TRUE if \$a is strictly greater than \$b.
\$a <= \$b	Less than or equal to	TRUE if \$a is less than or equal to \$b.
\$a >= \$b	Greater than or equal to	TRUE if \$a is greater than or equal to \$b.

Logical Operators

Example	Name	Result
\$a and \$b	And	TRUE if both \$a and \$b are TRUE.
\$a or \$b	Or	TRUE if either \$a or \$b is TRUE.
\$a xor \$b	Xor	TRUE if either \$a or \$b is TRUE, but not both.
! \$a	Not	TRUE if \$a is not TRUE.
\$a && \$b	And	TRUE if both \$a and \$b are TRUE.
\$a \$b	Or	TRUE if either \$a or \$b is TRUE.

References

Οι αναφορές στην PHP σας επιτρέπουν να κάνετε δυο μεταβλητές να αναφέρονται στο ίδιο περιεχόμενο. Δηλαδή, όταν γράφετε:

```
<?php
$a =& $b
?>
```

σημαίνει ότι η `$a` και η `$b` δείχνουν στην ίδια μεταβλητή. Η `$a` και η `$b` είναι τελείως ίσες εδώ, αφού η `$a` δεν δείχνει στην `$b` ή αντιστρόφως, αλλά τόσο η `$a` όσο και η `$b` δείχνουν στο ίδιο μέρος.

```
$a=5;
$b=$a;
$a=7 // $b is still 5
$b=&$a;
$a=10 // $b is 10 now
```

Ακύρωση ανάθεσης αναφορών

Όταν γίνεται ακύρωση της ανάθεσης με αναφορά, απλά διακόπτεται ο σύνδεσμος ανάμεσα στο όνομα της μεταβλητής και το περιεχόμενό της. Αυτό δε σημαίνει ότι το περιεχόμενο της μεταβλητής θα καταστραφεί. Για παράδειγμα:

```
<?php
$a = 1;
$b =& $a;
unset ($a);
?>
```

Το παραπάνω δεν θα ακυρώσει την ανάθεση στην `$b`, απλά στην `$a`.

Όπως ορίζει και το όνομά της η αναφορά στην `php` είναι μία μεταβλητή που “αναφέρεται” στα περιεχόμενα μίας άλλης μεταβλητής. Δείτε το παρακάτω παράδειγμα:

```
<?php
$foo = 5;
$bar = $foo;
for(int $i = 0; $i < 5; $i++)
{
    echo "The value of foo is: $foo, and the value of bar is: $bar<BR>";
    $foo++;
    $bar--;
}
?>
```

Όπως θα δείτε το αποτέλεσμα είναι 5 γραμμές κειμένου με την παρακάτω μορφή :

```
The value of foo is: 5, and the value of bar is 5
The value of foo is: 6, and the value of bar is 4
....
The value of foo is: 10, and the value of bar is: 0
```

Το κύριο σημείο είναι ότι οι μεταβλητές `$foo` και `$bar` είναι μεταξύ τους ανεξάρτητες και καθεμία μπορεί να αλλάξει χωρίς να επηρεάσει την άλλη. Η `php` επιτρέπει να δεθούν δύο ξεχωριστές μεταβλητές, επιτρέποντας έτσι την αλλαγή και των δύο ταυτόχρονα.

```
<?php
$foo = 5;
$bar =& $foo;
for(int $i = 0; $i < 5; $i++)
{
    echo "The value of foo is: $foo, and the value of bar is: $bar<BR>";
    $foo++;
    $bar--;
}
?>
```

Παρατηρήστε στη δεύτερη γραμμή ότι άλλαξε ο κώδικας. Αντί της απλής ανάθεσης της τιμής του `$foo` στην μεταβλητή `$bar`, χρησιμοποιήσαμε τον τελεστή `=&` για να θέσουμε

την μεταβλητή \$bar ως αναφορά στην μεταβλητή \$foo. Ποιά είναι η διαφορά; Ας δούμε την εκτέλεση :

```
The value of foo is: 5, and the value of bar is: 5
The value of foo is: 5, and the value of bar is: 5
....
The value of foo is: 5, and the value of bar is: 5
```

Όπως βλέπουμε καμία μεταβλητή δεν αλλάζει τιμή. Ας δούμε μία-μία τις γραμμές. Πρώτα αναθέτουμε την τιμή 5 στην μετ. \$foo. Κατόπιν δημιουργούμε την αναφορά στην μετ. \$foo και την καλούμε \$bar. Όταν μπαίνουμε στον κύκλο του for τυπώνουμε και των δύο μετ. τα περιεχόμενα που είναι 5. Κατόπιν αυξάνουμε το \$foo και μειώνουμε το \$bar. Γιατί όμως οι τιμές τους μένουν ίδιες ; Δείτε το παρακάτω σχήμα που επεξηγεί την διαφορά μεταξύ των δύο παραδειγμάτων...

Όπως βλέπουμε στο δεύτερο σχήμα είναι σαν να έχουμε μία μεταβλητή με δύο διαφορετικά ονόματα...

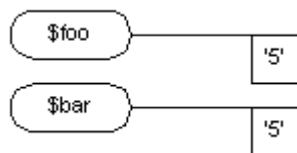


Figure 1: \$foo and \$bar as independent variables

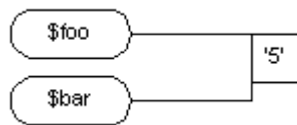


Figure 2: \$bar as a reference to the variable \$foo

Άρα όπως βλέπουμε στο δεύτερο παράδειγμα όταν η \$foo προσανξάνεται και κατόπιν η \$bar μειώνεται, λόγω της σχέσης αναφοράς με την \$foo μειώνεται έτσι και η \$foo με αποτέλεσμα να παραμένει πάντα η ίδια τιμή.

References and Arrays

Όπως και στις μεταβλητές έτσι η σχέση αναφοράς μπορεί να χρησιμοποιηθεί και στους πίνακες (arrays). Μία αναφορά στην php μπορεί να εφαρμοστεί σε ολόκληρο πίνακα, καθώς επίσης και σε ένα δείκτη πίνακα. Δείτε το παρακάτω παράδειγμα :

```
<?php
    $myarray = array(0=>'variable1', 1=>'variable2', 2=>'variable3');
    $ref1 = &myarray;    // Reference the entire Array
    $ref2 = &myarray[0]; // Reference to the first index
    $ref3 = &myarray[1]; // Reference to the second index
?>
```

Μετατροπές / Έλεγχος τύπου μεταβλητής

"42" είναι string, ενώ 42 είναι integer.

FALSE είναι boolean, ενώ "false" είναι string.

`gettype()`

μαθαίνουμε τον τύπο της μεταβλητής:

```
$number=5;
```

```
gettype($number)
```

`settype()`

ορίζουμε τον τύπο της μεταβλητής:

```
$number=5;
```

```
settype($number,"string")
```

Μπορεί να γίνει και με type casting όπως αναφέρθηκε και παραπάνω

`isset()`

Ελέγχει αν έχει οριστεί η μεταβλητή και επιστρέφει TRUE (ή τίποτα αν δεν έχει οριστεί)

```
echo isset($number)
```

`unset()`

Καταστρέφει μια variable

```
unset($number)
```

`empty()`

Το αντίθετο της `isset()`: επιστρέφει TRUE αν δεν έχει οριστεί, ή αν είναι "" ή 0 και τίποτα εάν έχει οριστεί η μεταβλητή.

`empty()` returns FALSE if var has a non-empty or non-zero value. In otherwords, "", 0, "0", NULL, FALSE, array(), var \$var;, and objects with empty properties, are all considered empty. TRUE is returned if var is empty.

`is_null()`

Μια variable θεωρείται NULL εάν της έχει αντιστοιχηθεί η constant NULL, εάν δεν της έχει οριστεί κάποια τιμή ή it has been [unset\(\)](#).

```
$var = NULL;
```

Returns TRUE if var is null, FALSE otherwise.

Loose comparisons with ==

	T	F	1	0	-1	"1"	"0"	"-1"	NULL	array()	"php"
T	T	F	T	F	T	T	F	T	F	F	T
F	F	T	F	T	F	F	T	F	T	T	F
1	T	F	T	F	F	T	F	F	F	F	F
0	F	T	F	T	F	F	T	F	T	F	T
-1	T	F	F	F	T	F	F	T	F	F	F
"1"	T	F	T	F	F	T	F	F	F	F	F
"0"	F	T	F	T	F	F	T	F	F	F	F
"-1"	T	F	F	F	T	F	F	T	F	F	F
NULL	F	T	F	T	F	F	F	F	T	T	F
array()	F	T	F	F	F	F	F	F	F	T	F
"php"	T	F	F	T	F	F	F	F	F	F	T

Strict comparisons with ===

	T	F	1	0	-1	"1"	"0"	"-1"	NULL	array()	"php"
T	T	F	F	F	F	F	F	F	F	F	F
F	F	T	F	F	F	F	F	F	F	F	F
1	F	F	T	F	F	F	F	F	F	F	F
0	F	F	F	T	F	F	F	F	F	F	F
-1	F	F	F	F	T	F	F	F	F	F	F
"1"	F	F	F	F	F	T	F	F	F	F	F
"0"	F	F	F	F	F	F	T	F	F	F	F
"-1"	F	F	F	F	F	F	F	T	F	F	F
NULL	F	F	F	F	F	F	F	F	T	F	F
array()	F	F	F	F	F	F	F	F	F	T	F
"php"	F	F	F	F	F	F	F	F	F	F	T

PHP 3.0 note: The string value "0" was considered non-empty in PHP 3, this behavior changed in PHP 4 where it's now seen as empty.

ΑΣΚΗΣΗ για variables

typical use→	variables usually with forms			variables usually with scripts			
↓test value→	something	12345	" "	0	FALSE	NULL	[unset]
!\$var	false	false	true	true	true	true	true
empty(\$var)	false	false	true	true	true	true	true
\$var==" "	false	false	true	true	true	true	true
\$var===0	false	false	true	false	false	false	false
isset(\$var)	true	true	true	true	true	false	false
strlen(\$var)==0	true	true	true	false	true	true	true
is_string(\$var)	true	false	true	false	false	false	false

This table demonstrates somewhat surprising results:

- For values like 0 (the integer), FALSE, and NULL, tests for falseness or emptiness like *!\$var* or *empty(\$var)* or *\$var==" "* will return true.
- A test like *strlen(\$var)==0* returns false when *\$var=0* (the integer) but true when *\$var=FALSE* or 12345. This is particularly confusing, since none of these variables is in fact a string, and two of them are equivalent in meaning even though they are different types.

What conclusions can we draw?

- When testing user input from a form, the **only** reliable test that the user has entered something is *if(\$var!="")*. But if you are expecting an array back, you need to test that it has been set, with *if(array())* (the *isset* test will not work with an array).
- When testing a conditionally set variable, you **must** check whether the variable has been set with *isset(\$var)* before any further processing, because some tests return true even with an unset variable.
- There are extremely subtle differences among these tests. These are often caused by mixing types in comparisons, like *if(strlen(\$var))* when *\$var=0* (the integer). Therefore, when testing a variable set by a script, you should consider carefully exactly which test will provide the results you want. Do strict rather than loose comparisons if possible. More than one test will probably be necessary.
- Always use the values TRUE/FALSE for boolean testing, never 1/0.
- For a function that is expected to return a value, consider setting a return value (*\$var=FALSE* or *\$var=""*) in case the function fails.

- The tests *if (!\$var)* and *if (empty(\$var))* and *if (\$var=="")* are completely interchangeable, and equally confusing. Each returns true for a variable that is empty or nonexistent in a general sense (either unset or ""), but also for a variable that is equal to 0 (which might be 0 the integer or FALSE the boolean, neither of which is empty or nonexistent in a general sense).

A metaphysical digression

- Analyzing exactly why a test like *if (!\$var)* returns true when *\$var=FALSE* gets us into metaphysics: can “nothing” nevertheless be something by virtue of being the thing “nothing”? To put it another way, if 0 and FALSE and NULL are in fact some sort of values, how can the variables containing them be considered false or empty? Apparently, in some cases they are recognized as not being empty:
- A test like *isset(\$var)* returns true for the values 0 (the integer) and FALSE, which are thus set even though supposedly they are nothing.
- A test like *strlen(\$var)==0* similarly returns false for the value 0 (the integer), which thus does have some length even though supposedly it is nothing.

HTML forms processing

HTML / web forms / Collecting Data

Σχεδιασμός φόρμας

Text box

Password Field

Textarea

Select / Option

Checkbox

Buttons

Radio Buttons

Hidden Fields

Μας ενδιαφέρει το METHOD (με ποιόν τρόπο θα αποσταλούν τα δεδομένα) και το ACTION (ποιο αρχείο θα επεξεργαστεί τα δεδομένα)

Upload Form

```
<form enctype="multipart/form-data" action="_URL_" method="post">
<input type="hidden" name="MAX_FILE_SIZE" value="30000">
Send this file: <input name="userfile" type="file">
<input type="submit" value="Send File">
</form>
```

Τρόποι αποστολής δεδομένων

- URL
- Forms (1.Controls 2.hidden fields)

Τρόποι αποστολής της φόρμας (METHOD):

- GET. Οι τιμές περνάνε σαν τμήμα του URL (π.χ. index.php?tmima=1&category=5), το query string. Το μειονέκτημα είναι πως μας περιορίζει στο μέγεθος που έχει το query string και επίσης είναι μη ασφαλές γιατί μπορεί να υποκλαπεί.
- POST. Το μειονέκτημα της POST είναι ότι δεν μπορεί να γίνει bookmarked, ωστόσο μπορούμε να περάσουμε δεδομένα χωρίς περιορισμό και είναι κάπως πιο ασφαλής από την GET (χωρίς αυτό να σημαίνει ότι είναι απολύτως ασφαλής μέθοδος).

Πέρασμα μεταβλητών και δεδομένων με HTML forms

Μέσα στο script, μπορούμε να έχουμε πρόσβαση στην τιμή κάθε control σαν να έχει οριστεί ως variable

Η register_globals επιτρέπει την δημιουργία μεταβλητών από τα HTML controls (π.χ. input type=text name="textbox32" θα δημιουργήσει μια variable μέσα στο PHP script με το όνομα textbox32.

Μια άλλη λύση είναι να χρησιμοποιήσουμε τα HTTP_GET_VARS ή / και HTTP_POST_VARS (ανάλογα με το αν το method είναι το GET ή το POST), καθώς και τις \$_GET και \$_POST (όταν είναι register_globals = off)

π.χ.

```
HTTP_GET_VARS["textbox32"];
$_GET["textbox32"];
```

Έστω ότι σχεδιάζουμε την παρακάτω φόρμα:

```
<form action="foo.php" method="POST">
  Name: <input type="text" name="username"><br>
  Email: <input type="text" name="email"><br>
  <input type="submit" name="submit" value="Submit me!">
</form>
```

Υπάρχουν 3 τρόποι:

```
<?php
// 1.
// Available since PHP 4.1.0
print $_POST['username'];
print $_REQUEST['username'];
// 2.
// Available since PHP 3. As of PHP 5.0.0, these long predefined
// variables can be disabled with the register_long_arrays directive.
print $HTTP_POST_VARS['username'];
// 3.
// Available if the PHP directive register_globals = on. As of PHP 4.2.0
// the default value of register_globals = off. Using/relying on this method is not preferred.
print $username;
?>
```

Tips for HTML, forms

- Επειδή πολύ συχνά ο HTML κώδικας περιέχει τα εισαγωγικά, θέλουμε να υπάρχει το \ για να αγνοούνται. Προκειμένου λοιπόν να μην το βάζουμε εμείς κάθε φορά που συναντάμε τα “”, όταν αποθηκεύουμε την HTML σελίδα, χρησιμοποιούμε την συνάρτηση addslashes:

```
$news=addslashes($news)
```

και όταν θέλουμε να εμφανίσουμε τη σελίδα μας, χρησιμοποιούμε την stripslashes:

```
$news=stripslashes($news)
```

addslashes

```
<?php
$str = "Is your name O'reilly?";
// Outputs: Is your name O\reilly?
echo addslashes($str);
?>
```

stripslashes

```
<?php
$str = "Is your name O'reilly?";
// Outputs: Is your name O'reilly?
echo stripslashes($str);
?>
```

- Χρήση του \$PHP_SELF στις φόρμες

Control Structures and Loops

IF, IF /ELSE, ELSEIF, Nested IF

Η δομή if είναι ένα από τα πιο σημαντικά χαρακτηριστικά σε πολλές γλώσσες, συμπεριλαμβανομένης και της PHP. Επιτρέπει την υπο συνθήκη εκτέλεση κομματιών κώδικα. Η PHP έχει μια δομή if παρόμοια με αυτή της C:

```
if (expr)
    statement

π.χ.
<?php
if ($a > $b)
{
    print "a is bigger than b";
    $b = $a;
}
?>
```

Συχνά θα θέλετε να εκτελέσετε μια δήλωση αν πληρείται μια συγκεκριμένη συνθήκη, και μια διαφορετική δήλωση αν αυτό δε συμβαίνει. Γι'αυτό το λόγο χρησιμοποιείται το else. Το else είναι επέκταση μιας δήλωσης if και εκτελεί μια δήλωση στην περίπτωση που η έκφραση στη δήλωση if είναι FALSE. Για παράδειγμα, ο ακόλουθος κώδικας θα εμφάνιζε a is bigger than b αν το \$a είναι μεγαλύτερο από το \$b, και a is NOT bigger than b στην αντίθετη περίπτωση:

```
<?php
if ($a > $b) {
    print "a is bigger than b";
} else {
    print "a is NOT bigger than b";
}
?>
```

Η εντολή elseif, όπως λέει και το όνομα της, είναι ένας συνδυασμός των if και else. Όπως το και else, έχει ως επέκταση μία if έκφραση με σκοπό να εκτελέσει μια διαφορετική έκφραση σε περίπτωση που η αρχική if συνθήκη πάρει την τιμή FALSE. Παρόλαυτα, σε αντίθεση με το else, θα εκτελέσει αυτή την εναλλακτική έκφραση μόνο αν η elseif υποθετική συνθήκη πάρει την τιμή TRUE. Για παράδειγμα, το ακόλουθο κομμάτι κώδικα θα εμφανίσει a is bigger than b, a equal to b or a is smaller than b:

```
<?php
if ($a > $b) {
    print "a is bigger than b";
} elseif ($a == $b) {
    print "a is equal to b";
} else {
    print "a is smaller than b";
}
?>
```

Μπορούν να υπάρχουν πολλά elseifs μέσα στην ίδια έκφραση if. Η πρώτη elseif έκφραση (αν υπάρχει) που θα πάρει την τιμή TRUE θα είναι και αυτή που θα εκτελεστεί. Στην PHP, μπορείτε επίσης να γράψετε 'else if' (σε δυο λέξεις) και η συμπεριφορά να είναι όμοια με αυτή του 'elseif' (μία λέξη). Η συντακτική έννοια είναι ελαφρώς διαφορετική (αν έχετε οικειότητα με τη C, είναι ακριβώς η ίδια συμπεριφορά) αλλά το τελικό αποτέλεσμα είναι ότι και οι δυο εκφράσεις θα καταλήξουν στην ίδια ακριβώς συμπεριφορά.

Η έκφραση elseif εκτελείται μόνο αν η προηγούμενη έκφραση if και οποιεσδήποτε προηγούμενες εκφράσεις elseif έχουν πάρει την τιμή FALSE, και η τρέχουσα έκφραση elseif πάρει την τιμή TRUE.

Οι δηλώσεις με If μπορούν να εμφωλευτούν απεριόριστα μέσα σε άλλες δηλώσεις if, κάτι το οποίο σας δίνει μεγάλη ευελιξία για την υπο συνθήκη εκτέλεση πολλών μερών του προγράμματος σας.

Ternary operator: condition ? value1 if true : value2 if false
(\$grade>50 ? "pass" : "fail");

Εναλλακτική σύνταξη για τις δομές ελέγχου (Alternative bracketing):

```
if (expression) :  
    statement block  
else :  
    statement block  
endif;
```

Η PHP προσφέρει μια εναλλακτική σύνταξη για ορισμένες από τις δομές ελέγχου της, δηλαδή για τις, if, while, for, foreach, και switch. Σε κάθε περίπτωση, η βασική μορφή της εναλλακτικής σύνταξης είναι η αλλαγή της παρένθεσης ανοίγματος με την άνω και κάτω τελεία (:) και της παρένθεσης κλεισίματος με τα endif;, endwhile;, endfor;, endforeach;, ή endswitch;, αντίστοιχα.

```
<?php if ($a == 5): ?>  
A is equal to 5  
<?php endif; ?>
```

Στο παραπάνω παράδειγμα, το μπλοκ της "A is equal to 5" είναι εμφωλευμένο μέσα σε μια έκφραση if γραμμένη με τον εναλλακτικό τρόπο σύνταξης. Το μπλοκ της HTML θα εμφανιστεί μόνο αν η \$a είναι ίση με 5.

Η εναλλακτική σύνταξη εφαρμόζεται και στο else και elseif . Η ακόλουθη δομή είναι μια δομή if με elseif και else με την εναλλακτική σύνταξη:

```
<?php  
if ($a == 5):  
    print "a equals 5";  
    print "...";  
elseif ($a == 6):  
    print "a equals 6";  
    print "!!!";  
else:  
    print "a is neither 5 nor 6";  
endif;  
?>
```

Η εναλλακτική σύνταξη ισχύει και για τις υπόλοιπες δομές ελέγχου, που αναλύονται παρακάτω.

Switch statement

Η δήλωση switch είναι παρόμοια με μια σειρά από IF δηλώσεις πάνω στην ίδια έκφραση. Σε πολλές περιπτώσεις, ίσως θέλετε να συγκρίνετε την ίδια μεταβλητή (ή έκφραση) με πολλές διαφορετικές τιμές, και να εκτελέσετε ένα διαφορετικό κομμάτι κώδικα ανάλογα με την τιμή με την οποία ισούται η μεταβλητή σας. Αυτό ακριβώς κάνει η δήλωση switch.

Σημείωση: Σημειώστε ότι σε αντίθεση με μερικές γλώσσες, η δήλωση [continue](#) εφαρμόζεται στην switch και ενεργεί παρόμοια με την break. Αν έχετε μια switch μέσα σε ένα loop και θέλετε να συνεχίσετε στην επόμενη επανάληψη του εξωτερικού loop, χρησιμοποιείτε την continue 2.

Τα ακόλουθα δυο παραδείγματα είναι δυο διαφορετικοί τρόποι για να γράψετε το ίδιο πράγμα, ο ένας χρησιμοποιεί μια σειρά από δηλώσεις if, και ο άλλος χρησιμοποιεί τη δήλωση switch :

```
<?php  
if ($i == 0) {  
    print "i equals 0";  
} elseif ($i == 1) {  
    print "i equals 1";  
} elseif ($i == 2) {  
    print "i equals 2";  
}
```

Είναι σημαντικό να καταλάβετε πώς η δήλωση switch εκτελείται προκειμένου να αποφευχθούν λάθη. Η switch εκτελείται γραμμή-γραμμή (για την ακρίβεια, δήλωση - δήλωση). Αρχικά δεν εκτελείται κάποιος κώδικας. Μόνο όταν μια δήλωση case βρεθεί με τιμή που ταιριάζει την τιμή της switch έκφρασης, η PHP αρχίζει να εκτελεί τις δηλώσεις. Η PHP συνεχίζει να εκτελεί τις δηλώσεις μέχρι το τέλος του switch block, ή την πρώτη φορά

που θα συναντήσει μια break δήλωση. Αν δεν γράψετε μια break δήλωση στο τέλος της λίστας δηλώσεων της case, η PHP θα συνεχίσει να εκτελεί τις δηλώσεις της επόμενης case. Για παράδειγμα:

```
<?php
switch ($i) {
    case 0:
        print "i equals 0";
    case 1:
        print "i equals 1";
    case 2:
        print "i equals 2";
}
?>
```

Εδώ, αν η \$i ισούται με 0, η PHP θα εκτελέσει όλες τις εντολές print! Αν η \$i ισούται με 1, η PHP θα εκτελέσει τις τελευταίες δύο εντολές print. Θα είχατε την αναμενόμενη συμπεριφορά ('η i ισούται με 2' θα εμφανιζόταν) μόνο αν η \$i ισούται με 2. Συνεπώς, είναι σημαντικό να μην ξεχνάτε τις εντολές break (ακόμη και αν θέλετε να αποφύγετε να τις τοποθετήσετε από προθεση κάτω από ορισμένες συνθήκες).

```
switch ($i)
{
    case 0:
        print "i equals 0";
        break;
    case 1:
        print "i equals 1";
        break;
    case 2:
        print "i equals 2";
        break;
}
?>
```

Σε μια switch εντολή, η συνθήκη υπολογίζεται μόνο μια φορά και το αποτέλεσμα συγκρίνεται με κάθε case εντολή. Σε μια elseif εντολή, η συνθήκη υπολογίζεται ξανά. Αν η συνθήκη σας είναι πιο περίπλοκη από μια απλή σύγκριση και/ή είναι σε ένα στενό loop, μια switch ίσως είναι ταχύτερη.

Η λίστα των δηλώσεων για μια case μπορεί επίσης να είναι άδεια, η οποία απλά περνά τον έλεγχο στη λίστα των δηλώσεων για την επόμενη case.

```
<?php
switch ($i) {
    case 0:
    case 1:
    case 2:
        print "i is less than 3 but not negative";
        break;
    case 3:
        print "i is 3";
}
?>
```

Μια ιδιαίτερη case είναι η default case. Αυτή η case ταιριάζει σε οτιδήποτε δεν ταιριάζουν οι άλλες cases, και θα πρέπει να είναι η τελευταία case εντολή. Για παράδειγμα:

```
<?php
switch ($i) {
    case 0:
        print "i equals 0";
        break;
    case 1:
        print "i equals 1";
        break;
    case 2:
        print "i equals 2";
        break;
    default:
        print "i is not equal to 0, 1 or 2";
}
```

```
}  
?>
```

Η έκφραση case μπορεί να είναι μια έκφραση η οποία υπολογίζεται σε έναν απλό τύπο, ο οποίος είναι ακέραιος (integer) ή κινητής υποδιαστολής (floating-point), αριθμοί ή γραμματοσειρές (strings). Πίνακες και αντικείμενα (objects) δεν μπορούν να χρησιμοποιηθούν εδώ, εκτός και αν αναχθούν σε έναν απλούστερο τύπο.

Η εναλλακτική σύνταξη για δομές ελέγχου υποστηρίζεται από τις switches.

```
<?php  
switch ($i):  
    case 0:  
        print "i equals 0";  
        break;  
    case 1:  
        print "i equals 1";  
        break;  
    case 2:  
        print "i equals 2";  
        break;  
    default:  
        print "i is not equal to 0, 1 or 2";  
endswitch;  
?>
```

For loops

Η σύνταξη ενός for loop είναι:

```
for (expr1; expr2; expr3) statement
```

Η πρώτη έκφραση (expr1) εκτελείται χωρίς να λάβουμε υπόψη κάποια συνθήκη στην αρχή του loop.

Στην αρχή κάθε επανάληψης, η expr2 υπολογίζεται. Αν πάρει την τιμή TRUE, το loop συνεχίζει και οι εμφωλευμένες εντολές εκτελούνται. Αν πάρει την τιμή FALSE, η εκτέλεση του loop σταματά.

Στο τέλος κάθε επανάληψης, υπολογίζεται η τιμή της expr3.

Κάθε μια από τις εκφράσεις μπορεί να είναι κενή. Αν η expr2 είναι κενή σημαίνει πως το loop θα εκτελείται ατέρμονα (η PHP αυτόματα θεωρεί πως έχει την τιμή TRUE, όπως και στη C).

Αυτό δεν είναι τόσο άχρηστο όσο φαίνεται αφού συχνά θέλουμε να τερματίζουμε ένα loop χρησιμοποιώντας μια σε υποθέση (condition) [break](#) δήλωση αντί να χρησιμοποιήσουμε την truth έκφραση της for .

Θεωρείστε τα ακόλουθα παραδείγματα. Όλα εμφανίζουν αριθμούς από το 1 ως το 10:

```
<?php  
/* example 1 */  
  
for ($i = 1; $i <= 10; $i++) {  
    print $i;  
}  
  
/* example 2 */  
for ($i = 1; ; $i++) {  
    if ($i > 10) {  
        break;  
    }  
    print $i;  
}  
  
/* example 3 */  
$i = 1;  
for (;;) {  
    if ($i > 10) {  
        break;  
    }  
    print $i;  
    $i++;  
}
```

```

}

/* example 4 */
for ($i = 1; $i <= 10; print $i, $i++);
?>

```

Φυσικά, το πρώτο παράδειγμα είναι το καλύτερο (ή ίσως το τέταρτο), αλλά ίσως δείτε πως έχοντας τη δυνατότητα να χρησιμοποιείτε κενές εκφράσεις στα for loops είναι πολύ εύχρηστο σε πολλές περιπτώσεις.

Η PHP επίσης υποστηρίζει την εναλλακτική "σύνταξη με χρήση της άνω και κάτω τελείας" στα for loops:

```

for (expr1; expr2; expr3): statement; ...; endfor;

```

Άλλες γλώσσες έχουν μια δήλωση foreach για να προσπελάσουν έναν array ή ένα hash. Η PHP 3 δεν έχει τέτοιες δομές. Η PHP 4 έχει (βλέπε foreach). Στην PHP 3, μπορείτε να συνδυάσετε το [while](#) με τη list() και την each() συνάρτηση προκειμένου να πετύχετε το ίδιο αποτέλεσμα.

While loops

Τα loops (βρόγχοι) while είναι ο απλούστερος τύπος loop στην PHP. Συμπεριφέρονται ακριβώς όπως και στη C. Η βασική μορφή μια δήλωσης while είναι η εξής:

```

while (expr)
{
    statements
}

```

Το νοήμα μιας δήλωσης while είναι απλό. Λέει στην PHP να εκτελέσει την εμφωλευμένη συνθήκη(ες) συνέχεια, μέχρι η έκφραση while να πάρει την τιμή TRUE. Η τιμή της έκφρασης ελέγχεται κάθε φορά στην αρχή του loop, έτσι ώστε ακόμη και αν αυτή η τιμή αλλάξει κατά τη διάρκεια της εκτέλεσης των εμφωλευμένων συνθηκών, η εκτέλεση δε θα σταματήσει μέχρι το τέλος της επανάληψης (κάθε φορά που η PHP εκτελεί τις εκφράσεις στο loop αποτελεί μια επανάληψη). Μερικές φορές, αν η έκφραση while πάρει την τιμή FALSE από την αρχή, η εμφωλευμένη εντολή-έκφραση δε θα εκτελεστεί ούτε μια φορά.

Όπως και στη δήλωση if, μπορείτε να βάλετε πολλές συνθήκες μέσα στο ίδιο while loop εσωκλείοντας τες μέσα σε { }, ή χρησιμοποιώντας την εναλλακτική σύνταξη:

```

while (expr): statement ... endwhile;

```

Τα παρακάτω παραδείγματα είναι όμοια, και τα δυο εκτυπώνουν τους αριθμούς από το 1 ως το 10:

```

<?php
/* example 1 */
$i = 1;
while ($i <= 10) {
    print $i++; /* the printed value would be
                $i before the increment
                (post-increment) */
}

/* example 2 */
$i = 1;
while ($i <= 10):
    print $i;
    $i++;
endwhile;
?>

```

Do...While

Το do..while loop είναι αρκετά όμοιο με το while loop, εκτός από το ότι η έκφραση αλήθειας ελέγχεται στο τέλος κάθε επανάληψης και όχι στην αρχή. Η κύρια διαφορά από τα κανονικά while loops είναι ότι η πρώτη επανάληψη ενός do..while loop εγγυάται την εκτέλεση του (η

αλήθεια της έκφρασης ελέγχεται μόνο στο τέλος της επανάληψης, ενώ μπορεί να μην είναι αναγκαίο να εκτελεστεί σε ένα κανονικό while loop (η αλήθεια της έκφρασης ελέγχεται στην αρχή κάθε επανάληψης, και αν πάρει την τιμή FALSE από την αρχή, η εκτέλεση του loop θα σταματήσει αμέσως).

Υπάρχει μόνο μια σύνταξη για τα do..while loops:

```
<?php
$i = 0;
do
{
    print $i;
} while ($i > 0);
?>
```

Το παραπάνω loop θα τρέξει ακριβώς μια φορά, αφού μετά την πρώτη επανάληψη, όταν η έκφραση αληθείας ελέγχεται, παίρνει την τιμή FALSE (το \$i δεν είναι μεγαλύτερο από το 0) και η εκτέλεση του loop σταματά.

Break

Η break τερματίζει την εκτέλεση της τρέχουσας εντολής for, foreach while, do..while ή της δομής switch .

Η break δέχεται ένα προαιρετικό αριθμό παραμέτρων ο οποίος της λέει πόσες εμφωλευμένες δομές πρέπει να διακοπούν-σπάσουν.

```
<?php
$arr = array ('one', 'two', 'three', 'four', 'stop', 'five');
while (list (, $val) = each ($arr)) {
    if ($val == 'stop') {
        break; /* You could also write 'break 1;' here. */
    }
    echo "$val<br>\n";
}

/* Using the optional argument. */
$i = 0;
while (++$i) {
    switch ($i) {
        case 5:
            echo "At 5<br>\n";
            break 1; /* Exit only the switch. */
        case 10:
            echo "At 10; quitting<br>\n";
            break 2; /* Exit the switch and the while. */
        default:
            break;
    }
}
?>
```

Continue

Η continue χρησιμοποιείται μέσα σε δομές loop προκειμένου να προσπεράσει το υπόλοιπο της επανάληψης του τρέχοντος loop και να συνεχίσει την εκτέλεση στην αρχή την επόμενης επανάληψης.

Η continue δέχεται έναν προαιρετικό αριθμό παραμέτρων ο οποίος της λέει πόσα επίπεδα εμπλεκόμενων loops πρέπει να προσπεράσει μέχρι το τέλος.

```
<?php
while (list ($key, $value) = each ($arr)) {
    if (!($key % 2)) { // skip odd members
        continue;
    }
}
```

```

    do_something_odd ($value);
}

$i = 0;
while ($i++ < 5) {
    echo "Outer<br>\n";
    while (1) {
        echo "&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;Middle<br>\n";
        while (1) {
            echo "&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;Inner<br>\n";
            continue 3;
        }
        echo "This never gets output.<br>\n";
    }
    echo "Neither does this.<br>\n";
}
?>

```

H continue δουλεύει με τα do, while, switch, for, foreach

Files, Directories and Email

File: μια ταξινομημένη ακολουθία από bytes αποθηκευμένη σε ένα κατάλληλο μέσο.

File operations (open, close, read, write, append)

Open file:

```
$fp=fopen("./file1.txt","r");  
if(!$fp) die("Cannot open file");
```

Η fopen() επιστρέφει ένα integer που είναι positive εάν κατάφερε να ανοίξει το αρχείο και 0 σε περίπτωση λάθους / αποτυχίας.

Το file μπορεί να είναι και URL.

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

Close file:

```
fclose($fp)
```

Πάντα είναι καλή πρακτική να κλείνουμε τα αρχεία.

Read a file:

```
$data=fread($fp,10)
```

θα διαβάσει τα 10 πρώτα bytes και θα προχωρήσει τον Pointer ανάγνωσης στην αρχή των επόμενων 10 bytes

```
<?php  
// get contents of a file into a string  
$filename = "/usr/local/something.txt";  
$handle = fopen($filename, "r");  
$contents = fread($handle, filesize($filename));  
fclose($handle);  
?>
```

Read the contents of a file

i. file_get_contents(\$filename) (only PHP 4.3.0 and above – ίσως ο καλύτερος τρόπος ανάγνωσης).

Για προγενέστερες εκδόσεις της PHP:

```
function file_get_contents($filename)  
{  
    $fd = fopen($filename, "rb");  
    $content = fread($fd, filesize($filename));  
    fclose($fd);  
    return $content;  
}
```

ii. file(): Ίδια με τη [readfile\(\)](#), εκτός του ότι η file() επιστρέφει τα αποτελέσματα σε array (κάθε element του array αποτελεί μια γραμμή του file). Σε περίπτωση σφάλματος,, επιστρέφει FALSE.

```
<?php  
// Get a file into an array. We'll go through HTTP to get the HTML source of a URL.  
$lines = file('http://www.example.com/');  
// Loop through our array, show html source as html source; and line numbers too.  
foreach ($lines as $line_num => $line) {  
    echo "Line #<b>{$line_num}</b> : " . htmlspecialchars($line) . "<br>\n";  
}  
// get a web page into a string. See also file_get_contents().  
$html = implode(" ", file('http://www.example.com/'));  
?>
```

iii. fpassthru(): διαβάζει και εκτυπώνει τα περιεχόμενα του αρχείου στον web browser.

```
$fp=fopen("file.txt","r");  
fpassthru=$fp;
```

iv. readfile(). Όμοια με την fpassthru, εκτός του ότι δεν απαιτεί έναν file handler:

```
readfile("file.txt");
```

Write to a file:

`fwrite($fp,$characters)` Returns the number of bytes that have been written and `-1` on failure of writing into the file.

1.1. File functions

Check the end of file:

`feof()`:

```
while (!feof($fp)
{ ... }
```

Read a string:

`fgets()`:

Διαβάζει lines μέχρι να βρεί χαρακτήρα `\n` (newline) ή να συναντήσει EOF ή να βρεί το τέλος του αριθμού των bytes που έχει οριστεί να διαβάσει

`$handle = fopen ("/tmp/inputfile.txt", "r");`

```
while (!feof ($handle)) {
    $buffer = fgets($handle, 4096);
    echo $buffer;
}
```

Read a character:

`fgetc()`: reads one character at a time. `$character=fgetc($fp)` είναι ίδιο με `$character=$fread($fp,1)`

`fileexists($file)`

`filesize($file)` size of file in bytes

`fileatime()` :accessed

`filectime()` :changed

`filemtime()` :modified

`is_writeable($filename)`

`is_dir($directory_name)`

Ownership – Permissions – Directories

Relative paths:

`./data.txt`: αναφέρεται στο ίδιο επίπεδο

`../data.txt`: αναφέρεται σε ένα επίπεδο πάνω

`../../data.txt`: αναφέρεται σε τρία επίπεδα πάνω

Absolute Paths:

Begin with `a /` . Αναφέρεται πάντα σε σχέση με το root directory του file system, όχι στη θέση του script.

Έλεγχος για την ύπαρξη καταλόγου: `is_dir($filename)`

Copy / Upload Files

`copy()`

`copy($source,$destination)`

`rename()`

`unlink()` explain the difference on unix / linux between linking and deleting on a physical media

Upload form in HTML

Include / require

Είναι το ίδιο, με τη διαφορά ότι σε περίπτωση αποτυχίας το require παράγει fatal error, ενώ το include παράγει warning.

Συνήθως χρησιμοποιούμε το require σε περιπτώσεις που το αρχείο είναι κρίσιμο για τη λειτουργία της εφαρμογής.

include 'file.txt'

Expression	gettype()	empty()	is_null()	isset()	boolean : if(\$x)	include ("file.txt")
\$x = "";	string	TRUE	FALSE	TRUE	FALSE	
\$x = NULL	NULL	TRUE	TRUE	FALSE	FALSE	
var \$x;	NULL	TRUE	TRUE	FALSE	FALSE	
\$x is undefined	NULL	TRUE	TRUE	FALSE	FALSE	
\$x = array();	array	TRUE	FALSE	TRUE	FALSE	
\$x = false;	boolean	TRUE	FALSE	TRUE	FALSE	
\$x = true;	boolean	FALSE	FALSE	TRUE	TRUE	
\$x = 1;	integer	FALSE	FALSE	TRUE	TRUE	
\$x = 42;	integer	FALSE	FALSE	TRUE	TRUE	
\$x = 0;	integer	TRUE	FALSE	TRUE	FALSE	
\$x = -1;	integer	FALSE	FALSE	TRUE	TRUE	
\$x = "1";	string	FALSE	FALSE	TRUE	TRUE	
\$x = "0";	string	TRUE	FALSE	TRUE	FALSE	
\$x = "-1";	string	FALSE	FALSE	TRUE	TRUE	
\$x = "php";	string	FALSE	FALSE	TRUE	TRUE	
\$x = "true";	string	FALSE	FALSE	TRUE	TRUE	
\$x = "false";	string	FALSE	FALSE	TRUE	TRUE	

Περιπτώσεις χρήσης:

- σε επαναλαμβανόμενες εργασίες με functions
- σύνδεση με βάση δεδομένων
- δόμηση / εμφάνιση της σελίδας (header, footer, css)

```
vars.php
<?php
$color = 'green';
$fruit = 'apple';
?>
```

```
test.php
<?php
echo "A $color $fruit"; // A
include 'vars.php';
echo "A $color $fruit"; // A green apple
?>
```

Όταν συμπεριλαμβάνεται ένα αρχείο, ο κώδικας που περιέχει κληρονομεί την εμβέλεια της μεταβλητής (variable scope) στη γραμμή στην οποία εμφανίζεται το include. Όποιες μεταβλητές είναι διαθέσιμες στη γραμμή εκείνη στο αρχείο που καλεί, θα είναι διαθέσιμες και μέσα στο καλούμενο αρχείο, από το σημείο εκείνο και μετά.

Η δήλωση include_once() περιλαμβάνει και υπολογίζει το συγκεκριμένο αρχείο κατά τη διάρκεια της εκτέλεσης του script. Αυτή η συμπεριφορά είναι όμοια με τη δήλωση include() , με τη μόνη διαφορά ότι αν ο κώδικας από ένα αρχείο έχει ήδη συμπεριληφθεί, δε θα ξανασυμπεριληφθεί. Όπως λέει και το όνομα του, θα συμπεριληφθεί μόνο μια φορά.

Η include_once() πρέπει να χρησιμοποιείται σε περιπτώσεις που το ίδιο αρχείο μπορεί να συμπεριληφθεί και να υπολογιστεί περισσότερες από μια φορές κατά την εκτέλεση ενός συγκεκριμένου script, και θέλετε να είστε σίγουροι ότι συμπεριλαμβάνεται ακριβώς μία φορά προκειμένου να αποφύγετε προβλήματα με επαναπροσδιορισμό συναρτήσεων, νέα ανάθεση τιμών στις μεταβλητές, κα.

Email

Το e-mail έχει 4 τμήματα: email address, subject, body, additional headers. Σαν header θεωρείται το CC, BCC, attachments, mime types κλπ και πρέπει να διαχωρίζονται με αλλαγή γραμμής (\n)

Send email:

mail(). Απαραίτητη προϋπόθεση να έχει οριστεί e-mail server στο php.ini (σε Unix, ο sendmail)

Mail attachments / MIME types:

Headers

Η συνάρτηση header() στέλνει απλούς HTTP headers.

Η header() ΠΡΕΠΕΙ να κληθεί πριν γίνει οποιοδήποτε output στον browser (HTML tags, blank lines ή από την PHP).

Caching

Συχνά οι browsers κάνουν caching τις σελίδες προκειμένου να τις φορτώνουν γρήγορα. Ωστόσο με την PHP και το δυναμικό περιεχόμενο γενικότερα δεν θέλουμε κάτι τέτοιο να συμβαίνει. Ο browser ελέγχει την ημέρα αλλαγής της σελίδας (modification time) προκειμένου να την κάνει caching. Αυτό σημαίνει ότι παρόλο που το php αρχείο δεν έχει αλλάξει, ο browser θα τοποθετήσει τη σελίδα στην cache – κάτι που δεν θέλουμε να συμβεί. Το περιεχόμενο της σελίδας μπορεί να έχει αλλάξει (π.χ. από τα Include αρχεία ή από τα στοιχεία της βάσης δεδομένων) και αν η σελίδα είναι cached, οι αλλαγές είναι πιθανόν να μην εμφανιστούν. Στέλνουμε επομένως κάποιο header στον browser για να αποφεύγεται το caching:

```
<?php
// Date in the past
// Expires: An old date, enforces the browser to check for a new version, thus not to cache
header("Expires: Mon, 26 Jul 1997 05:00:00 GMT");
// always modified
header("Last-Modified: " . gmdate("D, d M Y H:i:s") . " GMT");
// HTTP/1.1
header("Cache-Control: no-store, no-cache, must-revalidate");
header("Cache-Control: post-check=0, pre-check=0", false);
// HTTP/1.0
header("Pragma: no-cache");
?>
```

Άλλοι τρόποι χρησιμοποίησης της header()

- authentication (στέλνει header με μήνυμα λάθους)
- redirect σε άλλη σελίδα (π.χ. header('Location: <http://www.example.com/>'));
- εμφάνιση συγκεκριμένης σελίδας / αρχείου με τη χρήση των MIME types π.χ.
// We'll be outputting a PDF
[header\("Content-type: application/pdf"\);](#)

Τεχνικές / Πρακτικές για τη συγγραφή προγραμμάτων / scripts

- Χρήση σχολίων (comments). Βοηθάνε στην κατανόηση του κώδικα από άλλους developers
- Στοιχισμός του κώδικα. Βοηθάει στην καλύτερη ανάγνωση.
- Χρήση include files. Μειώνει το μέγεθος του κώδικα. Υπάρχει δυνατότητα επαναχρησιμοποίησης και άμεσων αλλαγών, χωρίς να αλλάζει ο βασικός κορμός του προγράμματος
- Χρήση functions. Προσφέρει καλύτερη δομή του κώδικα, επαναχρησιμοποίηση του κώδικα
- Αρχικοποίηση των μεταβλητών και χρήση ονομάτων με νόημα
- Κατά τη διάρκεια της ανάπτυξης μιας εφαρμογής, προσπαθήστε να διακόψετε την ομαλή λειτουργία με διάφορες δοκιμαστικές τιμές, προκειμένου να ανιχνεύσετε πιθανά λάθη.

Functions and Arrays

Χρησιμοποιούμε συναρτήσεις για:

- καλύτερη οργάνωση κώδικα
- επαναχρησιμοποίηση κώδικα (κερδίζουμε χρόνο και ο κώδικας είναι πιο αξιόπιστος)

Function (purpose, defining, calling)

```
function functionname (parameters)
{
    function code...
    return;
}
```

Χρησιμοποιώ το return σαν το τέλος της function:

```
function test($x1,$x2)
{
    $x=$x1+$x2
    return $x;
}
```

Τα parameters είναι values ή variables από τον user / program και χωρίζονται με κόμμα. Επίσης καταστρέφονται μετά τη λήξη της χρήσης της function

Ένα δεν θέλετε να περάσετε συγκεκριμένα arguments, δεν μπορείτε να χρησιμοποιήσετε τα κενά.

Ωστόσο, μπορείτε να θέσετε κενές τιμές (null values) σ' αυτές τις θέσεις.

Λάθος: \$x= fName(,,\$c);

Σωστό: \$x = fName(null,null,\$c);

Οι functions δεν εκτελούνται ποτέ, εκτός και αν κληθούν (να εκτελεστούν).

Δεν έχει σημασία σε ποιο τμήμα του προγράμματος θα γραφτεί η function.

Τα PHP function names δεν είναι case sensitive. Αυτό σημαίνει ότι τα fName(), FName(), and FNAME() αναφέρονται στην ίδια συνάρτηση.

Μπορώ επίσης να έχω nested functions. Ωστόσο χρειάζεται προσοχή στη σειρά κλήσης τους

```
<?php
function foo()
{
    function bar()
    {
        echo "I don't exist until foo() is called.\n";
    }
}

/* We can't call bar() yet since it doesn't exist. */
foo();

/* Now we can call bar(), foo()'s processing has made it accessible. */
bar();
?>
```

Επίσης μια function επιστρέφει πάντα μία τιμή. Δεν μπορούν να σας επιστραφούν πολλές τιμές από μια συνάρτηση, αλλά παρόμοια αποτελέσματα μπορούν να επιτευχθούν επιστρέφοντας ένα πίνακα.

```
<?php
function small_numbers()
{
    return array (0, 1, 2);
}
list ($zero, $one, $two) = small_numbers();
?>
```

Pass Arguments to a function (by value – by reference – by default value)

1.By value: εάν αλλάξουμε την τιμή μιας μεταβλητής / παραμέτρου μέσα στη συνάρτηση, αυτό δε σημαίνει ότι θα αλλάξει και έξω από τη συνάρτηση. Απλά δημιουργεί ένα copy της μεταβλητής που το χρησιμοποιεί

2.By reference: Εάν θέλουμε να αλλάξουμε την τιμή και έξω από τη συνάρτηση, πρέπει να γίνεται με αναφορά (passing by reference). Χρησιμοποιούμε το & πριν το όνομα της μεταβλητής / παραμέτρου:

```
function func1(&$var1)
{
    return $var1;
}
```

Μπορείτε να περάσετε μεταβλητές σε συναρτήσεις με αναφορά, έτσι ώστε η συνάρτηση να μπορεί να τροποποιήσει τα ορίσματα της. Η σύνταξη έχει ως ακολούθως:

```
<?php
function foo (&$var)
{
    $var++;
}
```

```
$a=5;
foo ($a);
// $a is 6 here
?>
```

Σημειώστε ότι δεν υπάρχει κάποιο σημάδι αναφοράς στην κλήση της συνάρτησης - μόνο στον ορισμό της συνάρτησης. Ο ορισμός της συνάρτησης από μόνος του είναι αρκετός για να περάσει σωστά τα ορίσματα με αναφορά.

Οι μεταβλητές, ουσιαστικά, είναι δείκτες (pointers) που δείχνουν σε τιμές που βρίσκονται στη μνήμη. Με την pass by value, δημιουργούμε ένα νέο pointer που δείχνει σε μια νέα τοποθεσία στη μνήμη που περιέχει μια κópια της κανονικής / αρχικής μεταβλητής. Αντίθετα με την pass by reference, δημιουργείται ένας νέος pointer που δείχνει όμως ακριβώς στην ίδια τοποθεσία μνήμης. Ουσιαστικά δεν περνάει την τιμή (value) πουθενά, απλά δημιουργεί νέες μεταβλητές που δείχνουν στην ίδια ακριβώς τοποθεσία. Με απλά λόγια δημιουργεί ένα alias αντί για μια κópια.

Επίσης δείτε το παρακάτω παράδειγμα:

Right: \$x = fName(&\$a, &\$b);

Wrong: fName(&\$a, &\$b);

3.Default Value:

```
<?php
function makecoffee ($type = "cappuccino")
{
    return "Making a cup of $type.\n";
}
echo makecoffee ();
echo makecoffee ("espresso");
?>
```

The output is:

Making a cup of cappuccino.

Making a cup of espresso.

Variable scope (global και local variables μέσα σε functions)

Local scope of variable:Μια variable είναι **local** όταν χρησιμοποιείται μέσα στον κορμό της function:

```
$x=5;
function printx()
{
```

```

    $x=1;
    echo "x inside the function is $x" . "<br>";
}
printx();
echo $x;

```

- 1^{ος} τρόπος: να δηλωθεί μέσα στο body της function ή ίδια μεταβλητή με τη λέξη **global**

```

$x=5;
function printx()
{
    global $x;
    ++$x;
    echo "x inside the function is $x" . "<br>";
}
printx();
echo $x;

```

- 2^{ος} τρόπος να χρησιμοποιήσω τον array GLOBALS: \$GLOBALS("var1")

```

<?php
$x=5;
function printx()
{
    ++$GLOBALS["x"];
    echo "x inside the function is " . $GLOBALS["x"] . "<br>";
}
printx();
echo $x;

```

Όταν θέλω μια μεταβλητή να κρατιέται πάντα σταθερή μέσα στην function, ύστερα από απανωτά καλέσματα της function, χρησιμοποιώ τη λέξη static: static \$var1. με τη χρήση της static η μεταβλητή συνεχίζει να διατηρεί την τιμή της και μετά το τέλος της function (μόνο όμως μέσα στον κορμό της function).

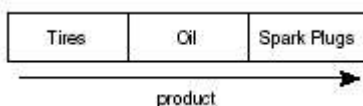
<p>Χωρίς τη χρήση static</p> <pre> function Test () { \$a = 0; echo \$a; \$a++; } Test(); Test(); Output:00 </pre>	<p>Με τη χρήση static</p> <pre> function Test() { static \$a = 0; echo \$a; \$a++; } Test(); Test(); Output:01 </pre>
--	---

Nested Functions

Recursion: a function calling itself

Arrays

Ένα σύνολο από μεταβλητές που έχουν όλες το ίδιο όνομα, αλλά διαφορετικό δείκτη (index). Κάθε στοιχείο του πίνακα ονομάζεται element.



```
$product=array("tires","oil","spark plugs");
```

Το γέμισμα του πίνακα γίνεται με τη συνάρτηση array (όπως το παραπάνω παράδειγμα).

Επίσης μπορώ να προσθέσω ένα στοιχείο στο τέλος του πίνακα με τη μορφή:

```
$product[]=$"new element";
```

Ordered Arrays:

```
$test=array(5,10,12,2);  
$test[2]=30;
```

Associative arrays:

```
$capital["skg"]="Thessaloniki";  
και το γέμισμα του associative array, γίνεται πάλι με τη χρήση της array() και του κατάλληλου  
κλειδιού / δείκτη (index key):
```

```
$capital = array('GRE'=>'Athens','GBR'=>'London');
```

Επίσης μπορώ να προσθέσω ένα στοιχείο στο τέλος του πίνακα με τη μορφή:

```
$capital["FRA"]="Paris";
```

Επανεγγραφή στοιχείου:

```
$capital["GRE"]="Thessaloniki";
```

Πρέπει πάντα να χρησιμοποιείτε εισαγωγικά έξω από ένα ευρετήριο ενός associative array.
Για παράδειγμα, χρησιμοποιήστε την \$foo['bar'] και όχι την \$foo[bar].

Κάθε string είναι ένα array:

```
$testing = "Hello";  
echo $testing[1];  
// output: e
```

Διαγραφή ενός στοιχείου: unset

```
unset( $example[3]);
```

implode: implode("delimiter",\$ArrayName) ένωση όλων των τιμών ενός array

```
<?php  
$array = array('lastname', 'email', 'phone');  
$comma_separated = implode(", ", $array);  
print $comma_separated; // lastname,email,phone  
?>
```

explode: \$ArrayName=explode("delimiter",\$stringname) τοποθετεί στον πίνακα τα substrings από το string που χωρίζονται με το delimiter.

```
<?php  
// Example 1  
$pizza = "piece1 piece2 piece3 piece4 piece5 piece6";  
$pieces = explode(" ", $pizza);  
print $pieces[0]; // piece1  
print $pieces[1]; // piece2  
  
// Example 2  
$data = "foo*:1023:1000::/home/foo:/bin/sh";  
list($user,$pass,$uid,$gid,$gecos,$home,$shell) = explode(":",$data);  
print $user; // foo  
print $pass; // *  
?>
```

Αντιστοίχιση στοιχείων / τιμών ενός πίνακα σε μεταβλητές – list

```
<?php  
$info = array('coffee', 'brown', 'caffeine');  
// Listing all the variables  
list($drink, $color, $power) = $info;  
print "$drink is $color and $power makes it special.\n";  
?>
```

Multidimensional Arrays

```
ArrayName = array (index=>array(ArrayContents))
```

	Code	Description	Price
product ↓	TIR	Tires	100
	OIL	Oil	10
	SPK	Spark Plugs	4
	product attribute →		

```
$products = array( array("TIR","Tires",100),
                   array("OIL","Oil",10),
                   array("SPK","Spark Plugs",4))
```

Foreach και while/list/each loops: επαναληπτικές δομές για πίνακες

while / list / each:

```
while (list(IndexValue,ElementContents) = each(ArrayName))
```

```
{ ... }
```

αν θέλουμε μόνο τα indexes: while (list(IndexValue) = each(ArrayName))

αν θέλουμε μόνο τα values: while (list(ElementContents) = each(ArrayName))

```
<?php
$fruit = array('a' => 'apple', 'b' => 'banana', 'c' => 'cranberry');
while (list($key, $val) = each($fruit)) {
    echo "$key => $val\n";
}
```

```
/* Outputs:
a => apple
b => banana
c => cranberry
*/
?>
```

foreach:

```
foreach ($ArrayName as $ArrayItem)
{ ... }
foreach ($ArrayName as $ArrayIndexValue => $ArrayItem)
{ ... }
```

Στην PHP 4 (και όχι στην PHP 3) συμπεριλαμβάνεται η δομή foreach, όπως στην Perl και σε μερικές άλλες γλώσσες. Αυτό απλά δίνει έναν εύκολο τρόπο να προσπελαύνετε τους πίνακες (arrays). Η foreach χρησιμοποιείται μόνο με πίνακες, και θα εμφανιστεί λάθος αν προσπαθήσετε να τη χρησιμοποιήσετε σε μια μεταβλητή διαφορετικού τύπου ή σε μια μεταβλητή που δεν έχει αρχικοποιηθεί. Υπάρχουν δυο τρόποι σύνταξης. Ο δεύτερος είναι μια ελάχιστη αλλά πολύ σημαντική βελτίωση του πρώτου:

```
foreach (array_expression as $value) statement
foreach (array_expression as $key => $value) statement
```

Η πρώτη φόρμα προσπελαύνει τον array σύμφωνα με την array_expression. Σε κάθε loop, η τιμή του τρέχοντος στοιχείου ανατίθεται στην \$value και ο εσωτερικός δείκτης του πίνακα αυξάνεται κατά ένα (έτσι ώστε στο επόμενο loop, θα μπορείτε να δείτε το επόμενο στοιχείο). Η δεύτερη φόρμα κάνει το ίδιο πράγμα, εκτός από το ότι το τρέχον στοιχείο του κλειδιού θα ανατίθεται στη μεταβλητή \$key σε κάθε loop.

Παρατηρήσεις:

- Όταν η foreach αρχίζει αρχικά να εκτελείται, ο εσωτερικός δείκτης του πίνακα αυτόματα πηγαίνει στο πρώτο στοιχείο του πίνακα. Αυτό σημαίνει ότι δεν χρειάζεται να καλέσετε τη συνάρτηση [reset\(\)](#) πριν από κάθε foreach loop.
- Επίσης, σημειώστε ότι η foreach εκτελείται σε ένα αντίγραφο του συγκεκριμένου πίνακα, όχι στον ίδιο τον πίνακα, συνεπώς ο δείκτης του πίνακα δεν αλλάζει όπως με τη δομή [each\(\)](#) και αλλαγές στο στοιχείο του πίνακα που επιστρέφεται δεν επηρεάζουν τον αρχικό πίνακα.

Παρόλαυτα, ο εσωτερικός δείκτης του αρχικού πίνακα *αυξάνεται* καθώς επεξεργαζόμαστε τον πίνακα. Υποθέτοντας ότι το foreach loop ολοκληρώνεται, ο εσωτερικός δείκτης του πίνακα θα είναι στο τέλος του πίνακα.

Ίσως έχετε παρατηρήσει ότι τα ακόλουθα έχουν την ίδια λειτουργία:

```
<?php
$arr = array("one", "two", "three");
reset ($arr);
while (list(, $value) = each ($arr)) {
    echo "Value: $value<br>\n";
}

foreach ($arr as $value) {
    echo "Value: $value<br>\n";
}
?>
```

Τα επόμενα λειτουργούν επίσης όμοια:

```
<?php
reset ($arr);
while (list($key, $value) = each ($arr)) {
    echo "Key: $key; Value: $value<br>\n";
}

foreach ($arr as $key => $value) {
    echo "Key: $key; Value: $value<br>\n";
}
?>
```

Μερικά ακόμη παραδείγματα για να καταλάβετε τη χρήση:

```
<?php
/* foreach example 1: value only */
$a = array (1, 2, 3, 17);
foreach ($a as $v) {
    print "Current value of \$a: $v.\n";
}

/* foreach example 2: value (with key printed for illustration) */
$a = array (1, 2, 3, 17);
$i = 0; /* for illustrative purposes only */
foreach ($a as $v) {
    print "\$a[$i] => $v.\n";
    $i++;
}

/* foreach example 3: key and value */
$a = array (
    "one" => 1,
    "two" => 2,
    "three" => 3,
    "seventeen" => 17
);
foreach ($a as $k => $v) {
    print "\$a[$k] => $v.\n";
}

/* foreach example 4: multi-dimensional arrays */
$a[0][0] = "a";
$a[0][1] = "b";
$a[1][0] = "y";
$a[1][1] = "z";
foreach ($a as $v1) {
    foreach ($v1 as $v2) {
        print "$v2\n";
    }
}

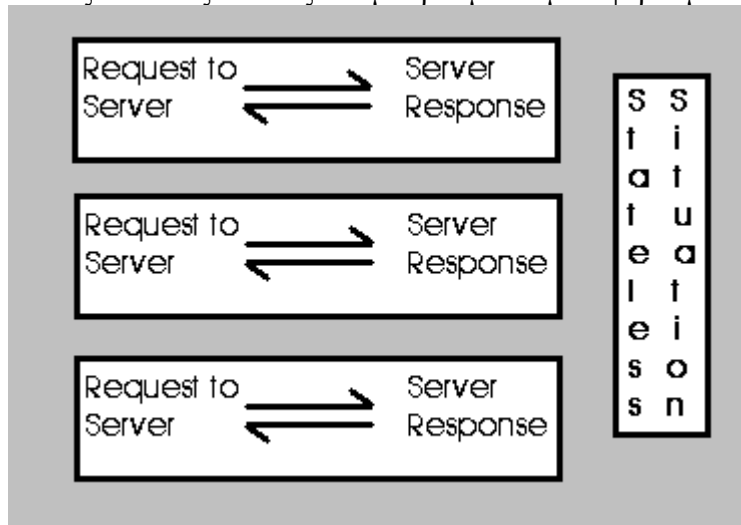
/* foreach example 5: dynamic arrays */
foreach (array(1, 2, 3, 4, 5) as $v) {
```

```
    print "$v\n";  
}  
?>
```

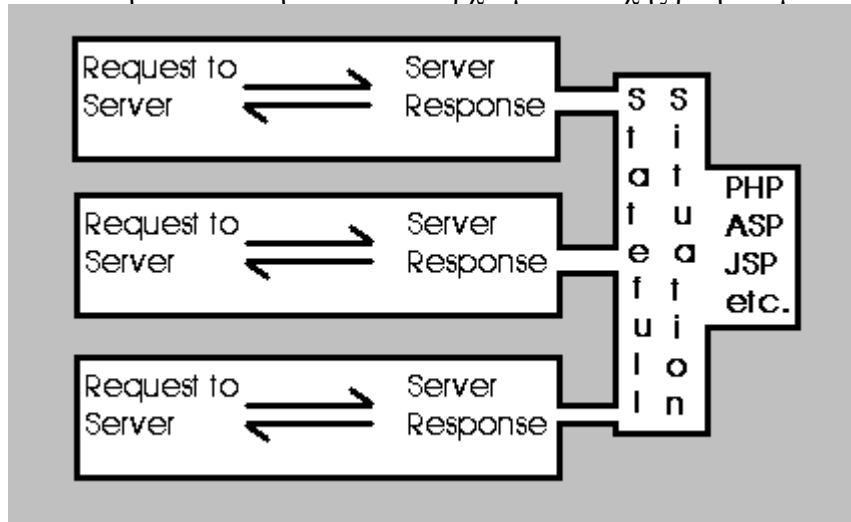
Cookies and Sessions

Describe persistence (γιατί την χρειαζόμαστε), hidden input fields to pass variables

Με τις στατικές σελίδες δεν μπορούμε να μεταφέρουμε πληροφορίες συνεχόμενα



Η ιδανική κατάσταση είναι να υπάρχει μια συνεχής ροή ανάμεσα στις σελίδες



Υπάρχει ανάγκη για sites με όσο το δυνατόν μεγαλύτερο βαθμό προσαρμογής σε προσωπικές επιλογές των χρηστών.

Ένας τρόπος να περνάνε κρυφά data από σελίδα σε σελίδα, είναι τα hidden input fields σε φόρμες.

Cookies

Τι είναι τα cookies

Η PHP έχει διαφανή υποστήριξη για HTTP cookies. Τα cookies είναι ένας μηχανισμός για αποθήκευση δεδομένων στον απομακρυσμένο browser και έτσι εντοπίζονται ή αναγνωρίζονται χρήστες που επιστρέφουν. Μπορείτε να ορίσετε cookies χρησιμοποιώντας την συνάρτηση setcookie(). Τα cookies είναι μέρος του HTTP header, έτσι η setcookie() πρέπει να καλεστεί πριν οποιαδήποτε έξοδος σταλεί στον browser. Αυτός είναι ο ίδιος περιορισμός που έχει η συνάρτηση header().

Οποιαδήποτε cookies στέλνονται σε σας από τον client θα μετατρέπονται αυτόματα σε PHP μεταβλητές όπως τα δεδομένα των GET και POST μεθόδων. Αν επιθυμείτε να καθορίσετε πολλές τιμές σε ένα μοναδικό cookie, απλά προσθέστε [] στο όνομα του cookie. Εάν ο client έχει περισσότερους από ένα web browser, κάθε browser θα διατηρεί το δικό του σύνολο από cookies

Που χρησιμοποιούνται:

- storing users preferences (how he wants the site to appear)
 - login purposes (last login, number of visits)
- ```
setcookie ("pass", $pass,time()+600); /* expires in 10 minutes */
setcookie ("user", $user,time()+600); /* expires in 10 minutes */
```

Τα cookies είναι καλά για μη κρίσιμες εργασίες ή για μη ευαίσθητα δεδομένα

### Create / read a cookie

```
setcookie():
 setcookie (name, value, expire, path, domain);
 e.g.
 setcookie("uname", $name, time()+3600)
```

### read / retrieve a cookie

```
if (isset($_COOKIE["uname"]))
 echo "welcome " . $_COOKIE["uname"];
else echo "not logged in";
```

### Delete a cookie

Δύο τρόποι:

1. reset the cookie only with its name
2. reset the cookie's expiry time to a time in the past

### Παράδειγμα με cookies

```
<?php
if(isset($bgcolor))
{
 setcookie("bgcolor", $bgcolor,time()+3600);
?>

<html>
<body bgcolor="#<?php echo $bgcolor; ?>"
<?php
}
else
{
 // display form
?>
WHAT IS YOUR FAVORITE COLOR?

<form name="color" action="<?php echo $PHP_SELF; ?>" method="post">
<select name="bgcolor">
 <option value="blue">BLUE</option>
 <option value="red">RED</option>
 <option value="green">GREEN</option>
</select>

<input type="submit" value="SET BGCOLOR">
</form>
<?php
}
?>
</body></html>
```

### Sessions

Ως session, ορίζουμε το διάστημα που μεσολαβεί από την πρώτη είσοδο του χρήστη στο site, έως και την έξοδο του. Κατά το διάστημα αυτό γίνονται διάφορες ενέργειες και μεταβολές και τα sessions μας επιτρέπουν τη διαχείριση όλων αυτών των ενεργειών.

Όταν ο χρήστης επισκέπτεται το web site, του αντιστοιχίζεται ένα μοναδικό id, το **session id (SID)**. Το SID στέλνεται στον browser του χρήστη σαν ένα cookie με όνομα PHPSESSID. Εάν τα cookies είναι απενεργοποιημένα, τότε το PHPSESSID επισυνάπτεται στο url, για όσο διαρκεί το session. Ταυτόχρονα, ένα αρχείο με το ίδιο όνομα δημιουργείται και στην πλευρά του server. Καθόλη τη διάρκεια του session, διάφορες μεταβλητές μπορούν να έχουν τιμές που αποθηκεύονται στο αρχείο του χρήστη.

enable-trans-sid: όταν κάνουμε compile την PHP με αυτό το flag, επισυνάπτει το PHPSESSID στο url

## Δημιουργία των sessions

session\_start()

Επιβάλλεται η χρήση της session\_start() function πρίνα από οτιδήποτε σταλθεί ως output στον browser. Σε κάθε άλλη περίπτωση θα δημιουργηθούν σφάλματα από την PHP.

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

Όταν ξεκινάει το session, συμβαίνουν τα εξής:

- αντιστοιχίζεται στο χρήστη το SID
- στέλνεται το cookie (εφόσον είναι ενεργή η ντιρεκτίβα session.use\_cookies στο php.ini)
- δημιουργείται το session file στον server

There's a problem with IE6: when you are using sessions and you post to a form, when you click your back button to make changes in the form, you have to click the REFRESH button on that page to get the information that you posted back into the form. This only works about 50% of the time, the other 50% the users information is lost and they have to type it over again. So, here's a solution for that: enter this right below the [session\\_start\(\)](#) of each script (yes it still must be before anything is output to the browser).

```
header("Cache-control: private");
```

Now your users can hit the back button and change information all they want!

```
<?php
session_start();
header("Cache-control: private"); // IE 6 Fix.
?>
```

Μια απλή html σελίδα:

```
<FORM METHOD="POST" ACTION="page2.php">
Enter your Name: <input type="text" name="name">
<input type="SUBMIT" value="Submit">
</FORM>
```

page2.php:

```
<?php
// start the session
session_start();
header("Cache-control: private"); //IE 6 Fix

echo "Step 2 - Register Session
";
// Get the user's input from the form
$name = $_POST['name'];
// Register session key with the value
$_SESSION['name'] = $name;
// Display the session information:
?>
Welcome to my website <? echo $_SESSION['name']; ?>!

Let's see what happens on the next page.


```

Το πρώτο που πρέπει να κάνουμε σε κάθε σελίδα που θα χρησιμοποιήσουμε session variable, είναι να ξεκινήσουμε το session..

page3.php:

```

<?php
// start the session
session_start();
header("Cache-control: private"); //IE 6 Fix
?>
Step 3 - Test Session Part II

Hey <? echo $_SESSION['name']; ?> Everything is still working!

Pick an option:

Let's delete this session value now. Click Here.

Let's destroy this session. Click Here.


```

Unregistering Session Variables:

```

$_SESSION['name'] = FALSE;
// or $_SESSION['name'] = "";

```

### **Διαγραφή του session:**

```
session_destroy();
```

Όταν διαγράφουμε το session, δεν σβήνονται τα cookies στον browser του χρήστη. Για να μην μπορούμε να τα χρησιμοποιήσουμε ύστερα από το τέλος του session, αρκεί να θέσουμε session.cookie\_lifetime = 0 στο php.ini

### **προβολή του session ID**

```

<?
session_start();
echo "Your session ID is ". session_id() . "";
?>
Your session ID is bd315d2ed59dfa1c2d0fb0b0339c758d

```

## Mysql and PHP

Ερωτήματα σε ΒΔ: προκειμένου να φιλτράρουμε τις πληροφορίες που μας χρειάζονται, να τις κατηγοριοποιήσουμε, να επιλέξουμε συγκεκριμένα δεδομένα, να εκτελέσουμε διάφορες λειτουργίες και ενέργειες στηριζόμενοι σε αυτά τα δεδομένα.

Χρησιμοποιούμε τα ερωτήματα για να εμφανίσουμε, να αλλάξουμε ή να αναλύσουμε δεδομένα με διαφορετικούς τρόπους.

Για το σκοπό αυτό πρέπει να χρησιμοποιήσουμε μια ειδική γλώσσα προγραμματισμού, την **SQL** (Structured Query Language – Δομημένη Γλώσσα Ερωτημάτων). Ο 2<sup>ος</sup> και εξίσου σημαντικός λόγος που χρησιμοποιούμε την SQL σε μια Βάση Δεδομένων είναι ότι μέσω των διαφόρων εντολών μπορούμε να δημιουργήσουμε τη δομή μιας βάσης δεδομένων (πίνακες, πεδία, εισαγωγή δεδομένων, δήλωση κλειδιών κ.λ.π.). Η SQL μπορεί να είναι λειτουργική και να δίνει καλά αποτελέσματα μόνο όταν έχω σχεδιάσει ικανοποιητικά τους πίνακες και τις σχέσεις μεταξύ τους.

Πιο συγκεκριμένα η sql μας επιτρέπει:

Ανάκτηση πληροφοριών με κριτήρια

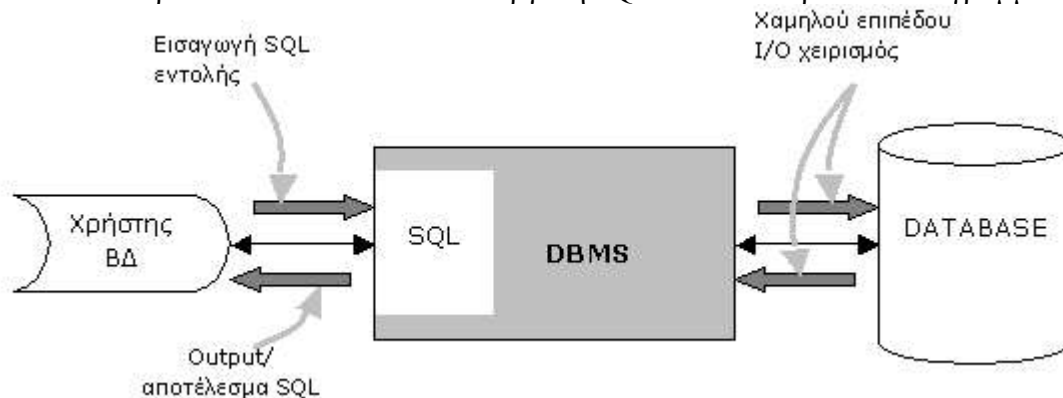
Διαγραφή δεδομένων

Ενημέρωση / καταχώρηση δεδομένων

Διαχείριση πινάκων

Διαχείριση χρηστών ΒΔ

Για να καταλάβετε σε ποιο επίπεδο λειτουργεί η SQL δείτε το παρακάτω διάγραμμα:



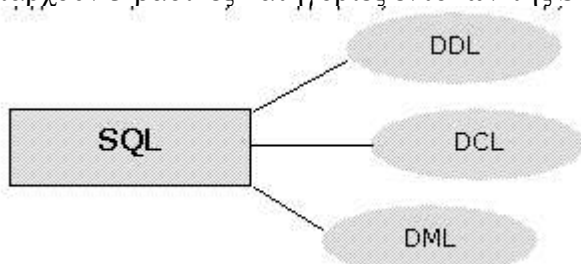
Ένα μεγάλο πλεονέκτημα που έχει η SQL είναι η αποδοχή της ως standard, ενώ αναγνωρίζεται σχεδόν από όλα τα λειτουργικά συστήματα και υπολογιστικά συστήματα (είναι cross-platform). Υπάρχουν 2 standards για την SQL, το ISO και το ANSI, με σχεδόν επικρατέστερο παντού το ANSI92 ή μεταγενέστερες εξελίξεις του.

Ο πιο συνηθισμένος τύπος ερωτήματος είναι αυτός που στηρίζεται στην εντολή SELECT. Ένα τέτοιο ερώτημα ανακτά πληροφορίες από ένα ή περισσότερα tables, χρησιμοποιώντας κριτήρια που ορίζει ο ενδιαφερόμενος και έπειτα εμφανίζει τα δεδομένα πάλι σύμφωνα με κριτήρια που ορίζει ο ενδιαφερόμενος (αύξουσα, φθίνουσα, αλφαβητική σειρά κ.λ.π.)

Αναλυτικότερη παρουσίαση των εντολών SQL γίνεται παρακάτω.

Κατηγορίες εντολών

Υπάρχουν 3 βασικές κατηγορίες εντολών της SQL:



**DDL (Data Definition Language).** Οι εντολές αυτής της κατηγορίας χρησιμοποιούνται για να ορίσουμε τη δομή μιας ΒΔ: δημιουργία, τροποποίηση και διαγραφή των πινάκων. Συνήθως οι DDL εντολές εκτελούνται από τον υπεύθυνο διαχείρισης του DBMS όταν έχουμε να κάνουμε με μια μεγάλη επιχείρηση. Σε λιγότερο απαιτητικές καταστάσεις, όπως μικρά projects και ΒΔ που δημιουργούνται από φοιτητές ή μεμονωμένους προγραμματιστές δεν υπάρχει περιορισμός στα δικαιώματα των εντολών αυτής της κατηγορίας. Συνήθεις DDL εντολές είναι οι CREATE TABLE, ALTER TABLE, DROP TABLE, CREATE VIEW, CREATE INDEX κλπ.

**DCL (Data Control Language).** Οι DCL εντολές είναι για γενικότερο έλεγχο πάνω στη ΒΔ: διαχείριση δικαιωμάτων των χρηστών, ανάκληση και επιβεβαίωση αλλαγών κλπ. Τυπικές DCL εντολές είναι οι GRANT, REVOKE, COMMIT, ROLLBACK κ.α.

**DML (Data Manipulation Language)** Αυτή η κατηγορία περιλαμβάνει εντολές για το χειρισμό των δεδομένων. Συνηθισμένες DML εντολές είναι οι INSERT, DELETE, UPDATE. Επίσης η κατηγορία αυτή περιλαμβάνει και τις εντολές SQL που χρησιμοποιούνται από κάποια γλώσσα προγραμματισμού (EXEC SQL, OPEN, close κ.α.)

Η mysql είναι ένα open source σχεσιακό DBMS με κύριο πλεονέκτημα την ιδιαίτερα μεγάλη ταχύτητα λειτουργίας. Επίσης δεν πρέπει να συγχέεται με την SQL (την γλώσσα προγραμματισμού που αναπτύχθηκε από την IBM): η mysql είναι ένα πρόγραμμα διαχείρισης ΒΔ που χρησιμοποιεί την SQL για να διαχειρίζεται και να εμφανίζει τα δεδομένα και τις πληροφορίες.

Τα πλεονεκτήματα της mysql είναι αρκετά:

- Ιδιαίτερα σταθερό, γρήγορο και αξιόπιστο DBMS
- Υποστηρίζει πλήρως το ANSI SQL92 standard
- Μπορεί να λειτουργήσει κάτω από σχεδόν οποιαδήποτε μεγάλη πλατφόρμα: Unix, Linux, Windows, MacOS, OS/2 κ.λ.π.
- Υποστηρίζει το πρωτόκολλο ODBC (παροχή σύνδεσης με άλλες πηγές δεδομένων), καθώς και πολλά ακόμη API για την σύνδεση με άλλα προγραμματιστικά περιβάλλοντα όπως C/C++ (mysql C API), PERL (mysql PERL API), Python και JDBC (παροχή σύνδεσης με την Java).
- Στις περισσότερες περιπτώσεις η απόκτηση της mysql είναι δωρεάν.
- Ιδιαίτερα υψηλή ασφάλεια και προστασία της Βάσης δεδομένων.

## Access MySQL using command mode / basic SQL syntax

Σύνδεση με τη mysql:

μπείτε στο mysql monitor με την εντολή mysql. Αυτή η εντολή σας επιτρέπει να συνδεθείτε σαν anonymous σε ένα server που λειτουργεί ως localhost. Για να βγείτε από τον mysql monitor πληκτρολογήστε quit.

```
mysql> QUIT
```

Bye

Άλλος τρόπος αποσύνδεσης είναι το \q ή το exit

Υπό κανονικές συνθήκες σε ένα δίκτυο και για λόγους ασφάλειας πρέπει να δώσετε username και password, ενώ σε περιπτώσεις που ο server είναι απομακρυσμένος (remote), πρέπει να δώσετε και όνομα host:

```
$> mysql -h host -u user -p
```

```
Enter password: *****
```

Το \*\*\*\*\* αναπαριστάει το password που δίνετε όταν σας το ζητάει η mysql

Εφόσον συνδεθείτε με επιτυχία θα εμφανιστούν κάποια μηνύματα:

```
$> mysql -u root -p
```

```
Enter password:
```

```
Welcome to the MySQL monitor. Commands end with ; or \g.
```

```
Your MySQL connection id is 157 to server version: 3.23.58-log
```

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.



Οι εντολές της mysql δεν είναι case sensitive, τα αντικείμενα (π.χ. πίνακες, πεδία, βάσεις δεδομένων) όμως που διαχειρίζεται είναι: διαφορετικό αποτέλεσμα έχει η εντολή create database dokimi; απ' ότι η εντολή create database Dokimi;

Για να δείτε ποιες databases υπάρχουν δώστε την εντολή show databases;

Με την εντολή use όνομα\_db λέτε στην Mysql ποια βδ πρόκειται να χρησιμοποιήσετε use mysql ή use test

Όταν δημιουργείτε μια νέα ΒΔ δεν σημαίνει απαραίτητα ότι θα την χρησιμοποιήσετε την παρούσα στιγμή. Πρέπει να δηλώσετε ποια θα χρησιμοποιήσετε και η εντολή use ουσιαστικά ενεργοποιεί την ΒΔ που πρόκειται να δουλέψετε.

Για να δείτε τη δομή μιας βδ δώστε την εντολή show tables from όνομα\_βδ;  
show tables from mysql;  
show tables from test;

Ένας εναλλακτικός τρόπος για να δείτε πάλι τη δομή της ΒΔ είναι από το shell του λειτουργικού συστήματος να δώσετε την εντολή mysqlshow όνομα\_βδ

Με την εντολή show columns from όνομα\_πίνακα; μπορείτε να δείτε τα πεδία ενός πίνακα από τη βδ που χρησιμοποιείτε:

show columns from user;

Εναλλακτική εντολή η describe όνομα\_πίνακα: describe user; ή desc user;

## Εργαλεία διαχείρισης MySQL

WinMySQL

PhpMyAdmin

## Βασικές εντολές SQL

### Δημιουργία μιας Βάσης Δεδομένων

Πριν δημιουργήσετε μια νέα βάση δεδομένων μπείτε μέσα στον mysql monitor (με την εντολή mysql)

Η δημιουργία μιας ΒΔ γίνεται με την εντολή create database όνομα\_βδ;

CREATE DATABASE sample\_db;

Για να μπορείτε τώρα να χρησιμοποιήσετε την νέα ΒΔ πρέπει να δώσετε την εντολή use όνομα\_βδ;

USE sample\_db;

### Διαγραφή μιας ΒΔ

Η διαγραφή μιας ΒΔ γίνεται με την εντολή drop database όνομα\_βδ;

DROP DATABASE sample\_db;

Πρέπει να σημειωθεί ότι όταν εκτελείτε την εντολή της διαγραφής μιας ΒΔ δεν υπάρχουν πλαίσια επιβεβαίωσης και επομένως πρέπει να είστε σίγουροι και ιδιαίτερα προσεκτικοί όταν την χρησιμοποιείτε. Η εντολή drop διαγράφει τη ΒΔ και τα περιεχόμενα της.

### Διαχείριση

Η διαχείριση στη mysql γίνεται με το σύστημα των δικαιωμάτων (privileges). Αυτό σημαίνει ότι ένας συγκεκριμένος χρήστης μπορεί να έχει πρόσβαση σε συγκεκριμένες βάσεις δεδομένων και μπορεί να εκτελεί συγκεκριμένες εντολές.

Υπάρχει μια ΒΔ στην mysql με το όνομα mysql που σας επιτρέπει να καταχωρείτε νέους χρήστες, έτσι ώστε να έχουν το δικαίωμα να χρησιμοποιούν τις ΒΔ. Αυτή η ΒΔ αποτελείται από 5 πίνακες: tables\_priv (δικαιώματα διαχείρισης πινάκων), columns\_priv (δικαιώματα

διαχείρισης πεδίων), user (καταχώρηση χρηστών), db (καταχώρηση ΒΔ) και host (καταχώρηση υπολογιστών-ποιοι hosts έχουν δικαίωμα πρόσβασης στον mysql server). Δεκτά / έγκυρα ονόματα για το host μπορεί να είναι ένα IP address (195.103.124.193), ένα πραγματικό hostname (mysql.com), το localhost. Ως localhost λειτουργούμε όταν ο server και ο client συνυπάρχουν στον ίδιο υπολογιστή.

### Δημιουργία πίνακα:

Η δημιουργία πίνακα (αν υποτεθεί ότι ήδη χρησιμοποιούμε μια ΒΔ), γίνεται με την εντολή:

```
create table table_name(όνομα στήλης1 τύπος δεδομένων,
...
...
...
όνομα στήληςN τύπος δεδομένων) ;
```

Επομένως έστω ότι έχουμε μια βάση test\_db (εάν δεν υπάρχει, δημιουργήστε την)

```
mysql>CREATE TABLE customers(
customer_id int not null primary key auto_increment,
first_name varchar(20) not null,
last_name varchar(30) not null,
address varchar(50),
city varchar(20),
state varchar(2),
zip varchar(5),
e_mail varchar(20),
age int,
race varchar(20),
gender enum('m','f'), default 'f');
```

Στην περίπτωση που δεν ξέρουμε εάν ένα table ήδη υπάρχει, χρησιμοποιούμε την εντολή:

```
create if not exists όνομα_πίνακα (όνομα στήλης1 τύπος δεδομένων,
...
...
...
όνομα στήληςN τύπος δεδομένων) ;
```

Κάποιοι κανόνες που αφορούν την δημιουργία και την ονομασία πεδίων στους πίνακες είναι χρήσιμο να τηρούνται: Χρησιμοποιείτε ονόματα που να περιγράφουν πραγματικά τη φύση των οντοτήτων και των γνωρισμάτων τους. Επίσης είναι καλή τακτική να περιγράφετε πλήρως ένα όνομα ή ένα χαρακτηριστικό χρησιμοποιώντας το \_ (π.χ customers\_table ή first\_name) – βοηθάει στο να περιγραφούν χωρίς παρερμηνίες.

### Null / Not Null

Το NULL είναι μια ειδική τιμή που μια στήλη/πεδίο μπορεί να πάρει. Το τι σημαίνει ακριβώς εξαρτάται από τον προγραμματιστή της βάσης δεδομένων: απύσα τιμή(missing value), δεν είναι γνωστή ακόμη(not yet known), μη εφαρμόσιμη (not applicable) ή κάτι άλλο.

Θα πρέπει να τονιστεί ότι το Null δεν έχει τη μαθηματική έννοια του όρου (δηλαδή 0). Παρόλα αυτά όταν ένας αριθμός συγκρίνεται με το NULL, το αποτέλεσμα είναι Null. (γενικότερα οποιαδήποτε αριθμητική πράξη εμπερικλείει το Null, το αποτέλεσμα είναι Null). Επίσης το Null δεν είναι ένας τύπος δεδομένων ή μια τιμή (value), αλλά παρόλα αυτά συμπεριφέρεται σαν εισαγωγή πεδίου (field entry): το 0 είναι μια τιμή, το Null δεν είναι. Το πρωτεύον κλειδί δεν μπορεί ποτέ να είναι null.

Όταν ένα πεδίο δηλώνεται ως not null, πρέπει οπωσδήποτε να εισάγουμε μια τιμή σε αυτό.

Επίσης θα πρέπει να σημειωθεί ότι το άδειο αλφαριθμητικό (" ") δεν είναι null -είναι not null.

Το ίδιο όπως προαναφέρθηκε ισχύει και για τις αριθμητικές αξίες (το 0 δεν θεωρείται null).

Για να γίνει πιο κατανοητό το θέμα εκτελέστε από την mysql το παρακάτω απλό ερώτημα:

```
mysql> SELECT 1 IS NULL, 1 IS NOT NULL;
```

Στη MySQL, 0 σημαίνει false  
σημαίνει true.

```
mysql> SELECT 1 IS NULL, 1 IS NOT NULL; και 1
+-----+-----+
| 1 IS NULL | 1 IS NOT NULL |
+-----+-----+
| 0 | 1 |
+-----+-----+
1 row in set (0.02 sec)
```

Επίσης δείτε και το επόμενο παράδειγμα

```
mysql> SELECT 1 = NULL, 1 != NULL, 1 < NULL, 1 > NULL;
mysql> SELECT 1 = NULL, 1 != NULL, 1 < NULL, 1 > NULL;
+-----+-----+-----+-----+
| 1 = NULL | 1 != NULL | 1 < NULL | 1 > NULL |
+-----+-----+-----+-----+
| 0 | 1 | 0 | 0 |
+-----+-----+-----+-----+
1 row in set (0.03 sec)
```

Όπως βλέπετε στο 2<sup>ο</sup> παράδειγμα το null δεν συμπεριφέρεται ως τιμή και επομένως δεν έχει νόημα η χρήση τελεστών όπως =, < ή !=.

Στη mysql αν ένα πεδίο δεν έχει καθοριστεί από το χρήστη ως null ή not null, αυτόματα του αποδίδεται η ιδιότητα null.

### Τύποι δεδομένων στη MySQL

Ως data type ορίζουμε τον τύπο δεδομένων που μπορεί να δεχθεί ένα πεδίο στον πίνακα μιας ΒΔ. Υπάρχουν 3 κατηγορίες δεδομένων: αριθμοί (numeric values), αλφαριθμητικά (strings) και η 3<sup>η</sup> αφορά τους υπόλοιπους τύπους (αντικείμενα, ημερομηνίες κ.λ.π.).

#### - Αριθμοί

Μπορούμε να συνοψίσουμε τις ιδιότητες των αριθμητικών τύπων στον παρακάτω πίνακα:

Type name	Memory space	Value range	unsigned
Tinyint	1 byte	-128 127	0-255
Smallint	2 bytes	-32768 32767	0-65535
Mediumint	3 bytes	-8388608 8388607	0-16777215
Int	4 bytes	-2147483648 2147483647	0-4294967295
Bigint	8 bytes	±9223372036854775808	
float(m,d)	4 bytes	Κυμαινόμενο	
Double(m,d)	8 bytes	Κυμαινόμενο	
Decimal(m,d)	τιμή του m +2 bytes	Κυμαινόμενο	

Παραδείγματα έγκυρων integers: 1221 ή 0 ή -32

Παραδείγματα έγκυρων floating-point αριθμών: 294.42 ή -32032.6809e+10 ή 148.00

Κάποιες χρήσιμες συμβουλές σχετικά με τους αριθμητικούς τύπους δεδομένων: είναι γενικός κανόνας ότι τα αριθμητικά δεδομένα επεξεργάζονται πιο γρήγορα από οποιονδήποτε άλλο τύπο δεδομένων. Επίσης χρησιμοποιείτε πεδία με αριθμητικό datatype σαν primary key ή indexing. Χρησιμοποιείτε τον τύπο decimal για χρηματικά δεδομένα προκειμένου να διατηρείτε την ακρίβεια στα ψηφία. Επίσης δώστε ιδιαίτερη προσοχή όταν μεταφέρετε βάση δεδομένων από ένα σύστημα σε κάποιο διαφορετικό: πρέπει να υπάρχει συμβατότητα ανάμεσα στους αριθμητικούς τύπους δεδομένων (και γενικότερα ανάμεσα στους τύπους δεδομένων).

Δηλώνοντας ένα πεδίο ως unsigned, είναι σαν να δηλώνετε την εισαγωγή σ' αυτό μόνο θετικών αριθμών: π.χ. ένας αριθμός δηλωμένος ως unsigned tinyint έχει εύρος από 0 έως 255. Αριθμοί τύπου float, double και decimal έχουν την δυνατότητα να κρατάνε και δεκαδικά ψηφία – οι άλλες κατηγορίες αριθμητικών τύπων δεν μπορούν. Ο αριθμός, για παράδειγμα 5.6876 θα αποθηκευτεί σε πεδίο που είναι δηλωμένο ως float(4,2) και συγκεκριμένα ως 5.68

#### - Αλφαριθμητικά

Το ΑΛΦΑΡΙΘΜΗΤΙΚΟ (STRING) είναι ένα σύνολο από χαρακτήρες. Πάλι πρέπει να δώσετε έμφαση, ανάλογα με τις ανάγκες αποθήκευσης, πόσο μικρό ή μεγάλο μέγεθος πρέπει να έχει ένα αλφαριθμητικό.

Type name	Μέγιστο μέγεθος	Storage space
Char(x)	255 bytes	X bytes
Varchar(x)	255 bytes	X+1 bytes
Tinytext	255 bytes	X+1 bytes
Tinyblob	255 bytes	X+2 bytes
Text	65535 bytes	X+2 bytes
Blob	65535 bytes	X+2 bytes
Mediumtext	1.6 MB	X+3 bytes
Mediumblob	1.6 MB	X+3 bytes
Longtext	4.2 GB	X+4 bytes
Longblob	4.2 GB	X+4 bytes

όπου X, το μήκος που δηλώνουμε

Οι πιο συνηθισμένοι αλφαριθμητικοί τύποι είναι οι char και varchar. Αλφαριθμητικά τύπου char χρησιμοποιούνται συνήθως για σταθερά μεγέθη, ενώ varchar για ευμετάβλητα. Εάν δηλώσετε αλφαριθμητικό char(10), τότε όλες οι τιμές θα δεσμεύουν και τα δέκα bytes, ακόμη και αν στην πραγματικότητα είναι μόνο 3 bytes. Εάν όμως δηλώσετε varchar(10) και αποθηκεύσετε ένα αλφαριθμητικό με μήκος 3 χαρακτήρες, τότε θα δεσμεύσει μόνο 4 bytes (μήκος αλφαριθμητικού+1) και όχι 10. Το παρακάτω παράδειγμα είναι χαρακτηριστικό:

Τιμή (Value)	Δήλωση ως CHAR(4)	Απαιτούνται	Δήλωση ως VARCHAR(4)	Απαιτούνται
"	' '	4 bytes	"	1 byte
'ab'	'ab '	4 bytes	'ab'	3 bytes
'abcd'	'abcd'	4 bytes	'abcd'	5 bytes
'abcdefgh'	'abcd'	4 bytes	'abcd'	5 bytes

Αλφαριθμητικά τύπου text και blob (binary large object) χρησιμοποιούνται για να αποθηκεύσουμε μεγάλα ποσά δεδομένων (εικόνες, ήχοι, ιστοσελίδες κ.λ.π.). Εάν αντιστοιχήσετε μια τιμή σε μια στήλη δηλωμένη ως blob ή text που υπερβαίνει το δηλωμένο μήκος, περικόπεται και ταιριάζει σε αυτό που ήδη δηλώσατε.

Κάποιες γενικές διευκρινήσεις σχετικά με τα αλφαριθμητικά:

Το αλφαριθμητικό δηλώνεται μέσα σε απλά ή διπλά εισαγωγικά, π.χ. 'a string' ή "another string". Επίσης, μέσα στο αλφαριθμητικό κάποιοι χαρακτήρες έχουν ειδικό νόημα / βαρύτητα.

Κάθε τέτοιος χαρακτήρας ξεκινάει με το backslash (\), γνωστός και ως escape character:

Χαρακτήρας	Επεξήγηση
\0	ASCII 0 (NUL) χαρακτήρας.
\n	Χαρακτήρας νέας γραμμής
\t	Χαρακτήρας tab
\r	Χαρακτήρας carriage return
\b	Χαρακτήρας backspace
\'	Συμπεριλαμβάνει απλό εισαγωγικό χαρακτήρα
\"	Συμπεριλαμβάνει απλό εισαγωγικό χαρακτήρα
\\	Χαρακτήρας backslash (\)
\% και \	Χαρακτήρας % και \

Κάποια σχετικά παραδείγματα:

```
mysql> SELECT 'hello', '"hello"', '""hello""', 'hel"lo', '\hello' as apotelesma;
```

```
+-----+-----+-----+-----+-----+
| hello | "hello" | ""hello"" | hel"lo | \hello |
+-----+-----+-----+-----+-----+
```

```
mysql> SELECT "hello", '"hello"', '""hello""', 'hel""lo', "\"hello" as printout;
```

```
+-----+-----+-----+-----+-----+
| hello | 'hello' | "hello" | hel"lo | "hello |
+-----+-----+-----+-----+-----+
```

```
mysql> SELECT "This\nIs\nFour\nlines" as ektiposi;
```

```
+-----+
| This
Is
Four
lines |
```

+-----+

#### - Διάφοροι τύποι

Ο τύπος ENUM σας επιτρέπει να δημιουργήσετε μια λίστα με τιμές, η οποία μπορεί να περιέχει μέχρι και 65535 ορίσματα. Π.χ. δήλωση τύπου `epilogi enum('y','n')` ή `age ('<18','18-25','25-44','44-65','>65')`.

Ο τύπος SET είναι παραπλήσιος με τον enum: οι μόνες διαφορές είναι ότι με τον set μπορείτε να αποθηκεύσετε περισσότερες της μιας τιμές και επίσης μπορούν να καταχωρηθούν στη λίστα μέχρι και 64 ορίσματα/τιμές. Παράδειγμα δήλωσης τύπου: `advertiser set ('TV','Newspaper')`. Σ' αυτή την περίπτωση αποδεκτές τιμές είναι οι " ", "TV", "Newspaper", "TV,Newspaper".

#### Τύποι ημερομηνίας και ώρας

Τύπος	Μορφή	Μηδενική τιμή (Zero value)
DATETIME	YYYY-MM-DD HH:MM:SS	0000-00-00 00:00:00
DATE	YYYY-MM-DD	0000-00-00
TIME	HH:MM:SS	00:00:00
YEAR	YYYY	0000
TIMESTAMP		

Το DATETIME χρησιμοποιείται όταν θέλουμε να αποθηκεύουμε μαζί και την ημερομηνία και την ώρα. Έγκυρες τιμές θεωρούνται αυτές μέσα στο εύρος '1000-01-01 00:00:00' έως '9999-12-31 23:59:59'.

Για να αποθηκεύσουμε μόνο την ημερομηνία χρησιμοποιούμε το DATE. Η MySQL εμφανίζει τις DATE τιμές σε μορφή 'YYYY-MM-DD'. Το υποστηριζόμενο εύρος είναι '1000-01-01' έως '9999-12-31'.

Το TIME αποθηκεύει το χρόνο με τη μορφή 'HH:MM:SS'. Δεκτές μορφές θεωρούνται οι 08:32:20, 08-32-20, 8.32.20 ακόμη και οι 083220 ή 83220 (χρειάζεται όμως πάντα προσοχή σε μια τέτοια περίπτωση γιατί η mysql διαβάζει την ημερομηνία από δεξιά προς τα αριστερά και επομένως μια δήλωση τύπου 832 θα εκληφθεί ως 32 sec και 8 min!).

Ο τύπος δεδομένων YEAR καταλαμβάνει 1 byte και η μορφοποίηση του μπορεί να είναι με 4 ψηφία ή με 2. Το εύρος είναι από το 1901 έως το 2155. Χρειάζεται ωστόσο προσοχή όταν δηλώνεται με 2 ψηφία γιατί οι τιμές από 00 έως 69 θεωρούνται τα έτη 2000-2069 και από 70 έως 99 θεωρούνται τα έτη 1970-1999. Μη έγκυρες τιμές για το YEAR μετατρέπονται σε 0000.

Το TIMESTAMP επιστρέφει την σημερινή ημερομηνία και ώρα ως καταχώρηση στο πεδίο που έχει δηλωθεί ως timestamp. Αυτό σημαίνει ότι όταν κάνετε μια καταχώρηση δεν είστε υποχρεωμένοι στο πεδίο αυτό να δώσετε μια τιμή – εισάγεται αυτόματα. Το timestamp μπορεί να δηλωθεί με τις ακόλουθες μορφές:

Timestamp δήλωση	Επεξήγηση
TIMESTAMP(14)	YYYYMMDDHHMMSS
TIMESTAMP(12)	YYMMDDHHMMSS
TIMESTAMP(10)	YYMMDDHHMM
TIMESTAMP(8)	YYYYMMDD
TIMESTAMP(6)	YYMMDD
TIMESTAMP(4)	YYMM
TIMESTAMP(2)	YY

Η δήλωση ενός πεδίου ως timestamp καταλαμβάνει 4 bytes. Εάν έχετε δηλώσει πολλαπλές στήλες ως τύπο TIMESTAMP, μόνο η πρώτη από αυτές θα ενημερωθεί αυτόματα.

Θα πρέπει επίσης να σημειωθεί ότι δεν μπορείτε να αφήσετε κενό –null ένα πεδίο που έχει δηλωθεί ως timestamp: άσχετα με το αν έχει δηλωθεί ως null ή not null, το πεδίο θα περιέχει ως τιμή την ημερομηνία/ώρα που έγινε η καταχώρηση. Επομένως η δήλωση null/not null δεν συμπεριφέρεται με τον κανονικό λογικό τρόπο και στην περίπτωση του timestamp αγνοείται. Ωστόσο αυτό δεν σας εμποδίζει να δηλώσετε ως περιεχόμενο αυτού του πεδίου το 0 – γιατί πολύ απλά το 0 δεν είναι null.

```
mysql> desc try_table;
```

Field	Type	Null	Key	Default	Extra
id	int(11)		PRI	NULL	auto_increment
ora	timestamp(12)	YES		NULL	

```
2 rows in set (0.00 sec)
```

```
mysql> select * from try_table;
```

id	ora
3	011112174245
4	011112174350
5	000000000000
6	011112181741

```
4 rows in set (0.00 sec)
```

Το παράδειγμα είναι αρκετά σχετικό – αν θέλατε να εισάγετε μια νέα εγγραφή δοκιμάστε την εντολή `insert into try_table values()`;

Η ιδιότητα `AUTO_INCREMENT` αυτόματα αυξάνει την τιμή σε όλους τους τύπους ακέραιων κατά 1. Πιο συγκεκριμένο, για κάθε νέα εγγραφή το πεδίο που έχει δηλωθεί ως `auto_increment` αυξάνει την αξία του κατά 1. Στον ορισμό ενός πίνακα και ενός πεδίου ως `auto_increment` μπορούμε να ξεκινήσουμε την αυτόματη αρίθμηση και από μια τιμή της επιλογής μας:

```
CREATE TABLE Orders (
 OrderID INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
 CustomerID VARCHAR(5),
 EmployeeID INT,
 OrderDate DATETIME)
 AUTO_INCREMENT=50;
```

Επίσης, επιτρέπεται η δήλωση μόνο μιας στήλης του πίνακα με την ιδιότητα `auto_increment` – όχι περισσότερες. Παρόλο που το πεδίο με την ιδιότητα `auto_increment` έχει οριστεί ως `not null`, του αντιστοιχίζεται ως `default` τιμή η `null` (προκειμένου να ξεκινήσει η αρίθμηση).

Ακόμη, σε στήλες με την ιδιότητα `auto_increment` πρέπει να αντιστοιχίζονται `indexes`.

Η ιδιότητα `ZEROFILL` χρησιμοποιείται για να εμφανίζει μηδενικά πριν από ένα αριθμό: αν δηλώσουμε έναν αριθμό ως `int(8) zerofill` και η τιμή που έχουμε αποθηκεύσει είναι 23, τότε θα εμφανιστεί ως 00000023.

Η ιδιότητα `DEFAULT` σας επιτρέπει την ύπαρξη μιας τιμής σε ένα πεδίο πριν ακόμη εισάγετε δεδομένα: δήλωση τύπου `enum('y','n') default 'n'`

Η ιδιότητα `PRIMARY KEY` καθορίζει στο πεδίο το πρωτεύον κλειδί:

**όνομα\_πεδίου δήλωση\_datatype primary key.**

Ένα παράδειγμα δίνεται με τον πίνακα `Orders`

```
CREATE TABLE Orders (
 OrderID INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
 CustomerID VARCHAR(5),
 EmployeeID INT,
 OrderDate DATETIME);
```

Επίσης μπορεί να γίνει με τον ακόλουθο τρόπο:

```
CREATE TABLE Orders (
 OrderID INT NOT NULL AUTO_INCREMENT,
 CustomerID VARCHAR(5),
 EmployeeID INT,
 OrderDate DATETIME,
 PRIMARY KEY(OrderID);
```

Πρέπει να σημειωθεί ότι σε κάθε πίνακα επιτρέπεται μόνο ένα πρωτεύον κλειδί (άσχετα με το αν αυτό αποτελείται από ένα ή περισσότερα πεδία). Επίσης η `mysql` απαιτεί η δήλωση ενός πεδίου ως `primary key` να συνοδεύεται και από την ιδιότητα `not null`.

Στην περίπτωση που είχαμε δημιουργήσει τον πίνακα και δεν είχαμε προσθέσει το πρωτεύον κλειδί, το διορθώνουμε με την εντολή:

ALTER TABLE όνομα\_πίνακα ADD PRIMARY KEY (πεδίο1,...);

Για να ορίσω ένα πρωτεύον κλειδί που αποτελείται από περισσότερα του ενός, απλά τα δηλώνω στην ιδιότητα primary key:

```
CREATE TABLE shop (
 article INT(4) UNSIGNED DEFAULT '0000' NOT NULL,
 dealer CHAR(20) DEFAULT '' NOT NULL,
 price DOUBLE(16,2) DEFAULT '0.00' NOT NULL,
 PRIMARY KEY(article, dealer));
```

Για να διαγράψουμε ένα πρωτεύον κλειδί χρησιμοποιούμε την εντολή:

```
alter table όνομα_πίνακα drop primary_key;
```

Η ιδιότητα UNIQUE κατοχυρώνει ότι όλα τα δεδομένα μέσα στο δηλωμένο πεδίο πρέπει να είναι μοναδικά και σε κάθε περίπτωση που προσπαθείτε να εισάγετε μια τιμή που δεν είναι μοναδική για το πεδίο, θα εμφανίζει λάθος. Ωστόσο, η mysql επιτρέπει σε ένα πεδίο ορισμένο ως unique να περιέχει null values. Η ιδιότητα unique είναι αρκετά σημαντική γιατί εκτός από λόγους απόδοσης της ΒΔ (είναι ιδιαίτερα αποτελεσματική στην αναζήτηση με τη χρήση του Where), εξασφαλίζει και την ακεραιότητα δεδομένων (data integrity) – κάθε τιμή είναι μοναδική.

## Select

Δημιουργούμε τον παρακάτω πίνακα για να εκτελέσουμε τα παραδείγματα:

```
CREATE TABLE shop (
 Article INT(4) UNSIGNED ZEROFILL DEFAULT '0000' NOT NULL,
 ArticleName CHAR(20) NOT NULL,
 salesman CHAR(20) DEFAULT '' NOT NULL,
 price DOUBLE(16,2) DEFAULT '0.00' NOT NULL,
 datesold DATE,
 PRIMARY KEY (ARTICLE));
```

και εισάγουμε τα παρακάτω δεδομένα:

article	articlename	salesman	price	datesold
0001	audi	A	3.45	2001-01-01
0002	Volkswagen Polo	B	3.99	2001-03-04
0003	Toyota Corolla	A	10.99	1999-10-12
0004	Volkswagen Golf	B	1.45	2000-05-08
0005	Toyota Yaris	C	6.70	1998-04-07
0006	Peugeot 206	D	1.25	2000-03-22
0007	Peugeot 206	D	19.25	1999-03-14

Η εντολή select μπορεί να χρησιμοποιηθεί για την εύρεση διαφόρων στοιχείων από την βάση δεδομένων

Για να δούμε την έκδοση της MySQL που χρησιμοποιούμε

```
mysql> SELECT VERSION();
```

Για την τρέχουσα ημερομηνία

```
mysql> SELECT CURRENT_DATE;
```

Για την ώρα: SELECT NOW();

Για να δούμε το όνομα του χρήστη που είναι συνδεδεμένος στην βάση

```
mysql> SELECT USER();
```

Ακόμη και για να υπολογίσουμε μια αριθμητική παράσταση

```
mysql> SELECT SIN(PI() /4), (4+1)*5;
```

Ωστόσο η κύρια χρήση της έχει να κάνει με την επιλογή δεδομένων. Η γενική μορφή σύνταξης της εντολής είναι:

```
SELECT what_to_select
FROM which_table
WHERE conditions_to_satisfy
```

Εμφάνιση όλων των δεδομένων των στηλών: SELECT \* FROM shop;

Εμφάνιση του πεδίου salesman: SELECT salesman FROM shop;

Το πεδίο ή τα πεδία που ορίζετε να εμφανιστούν μπορείτε να τα δώσετε ένα διαφορετικό όνομα στην εμφάνιση με τη χρήση του as .Το προηγούμενο παράδειγμα θα είναι:  
 SELECT salesman as pwlites FROM shop;  
 Επίσης για να μην επαναλαμβάνεται η εμφάνιση δεδομένων και να εμφανίζονται διάκριτες γραμμές: SELECT DISTINCT salesman FROM shop;

Για την επιλογή μόνο των εγγραφών με πωλητή τον Α

```
SELECT *
FROM shop
WHERE salesman="A";
```

Και για την επιλογή μόνο του πωλητή και της τιμής

```
SELECT salesman, price
FROM shop
WHERE price > 5;
```

Οι τελεστές που χρησιμοποιούμε μπορεί να είναι: > , >=, <, <=, <> ή <=

Οι συνθήκες που απαρτίζουν το where μπορούν να συνδυαστούν με πολλούς τρόπους.

Εδώ ισχύουν οι γνωστοί κανόνες της λογικής σύζευξης (AND) και λογικής διάζευξης (OR) καθώς και της λογικής άρνησης (NOT)

Πρόταση Α	Πρόταση Α	Α και Β	Α ή Β	NOT Α
Αληθής	Αληθής	Α	Α	Ψ
Αληθής	Ψευδής	Ψ	Α	Ψ
Ψευδής	Αληθής	Ψ	Α	Ψ
Ψευδής	Ψευδής	Ψ	Ψ	Α

Χρήση του AND

```
SELECT * FROM shop WHERE salesman="B" AND price > 3;
```

Χρήση του OR

```
SELECT * FROM shop WHERE salesman="A" OR salesman="B";
```

Συνδυαστική χρήση τελεστών

```
SELECT * FROM shop
where (salesman="A" and price >5) OR (salesman="B" and price>5);
```

Χρήση του NOT

```
SELECT * from shop
WHERE NOT price > 5;
```

Αξίζει να σημειωθεί ότι χρειάζεται προσοχή στη σύνταξη των λογικών συναρτήσεων και ιδιαίτερα στη σειρά με την οποία μπαίνουν οι παρενθέσεις, όπου αυτό χρειάζεται

Έχουμε την δυνατότητα να εμφανίζουμε τις εγγραφές με σειρά αύξουσα ως προς ένα πεδίο του πίνακα

```
SELECT article, price
FROM shop
ORDER BY price;
```

ή η ταξινόμηση με βάση την ημέρα πώλησης:

```
SELECT * FROM shop order by Datesold;
```

Ακόμη και με σειρά φθίνουσα με τη χρήση του desc

```
SELECT article, Price
FROM shop
ORDER BY price desc;
```

Με τη χρήση του Between μπορούμε να ψάχνουμε σε ένα εύρος τιμών.

```
SELECT * FROM shop
WHERE datesold BETWEEN '2000-1-1' AND '2002-1-1';
```

Όταν έχουμε να αναζητήσουμε μεταξύ διακριτών τιμών, αντί να χρησιμοποιήσουμε τη σύνταξη του OR, μπορούμε να κάνουμε χρήση του IN.

Αντι για παράδειγμα να γράψουμε

```
SELECT * FROM shop
WHERE salesman="A" OR salesman="B" OR salesman="C";
```

Μπορούμε να γράψουμε

```
SELECT * FROM shop
WHERE salesman IN ('A', 'B', 'C');
```



Η επόμενη αναζήτηση θα μας δώσει όλες τις εγγραφές που το articlename αρχίζει από 'Volks'. Το σύμβολο δηλαδή % επέχει τη θέση του «οτιδήποτε».

```
SELECT * FROM shop
WHERE articlename LIKE "Volks%";
```

Η επόμενη αναζήτηση θα μας δώσει όλες τις εγγραφές που το articlename τελειώνει σε 'a'.

```
SELECT * FROM shop
WHERE articlename LIKE "%a";
```

Ενώ η επόμενη οτιδήποτε περιέχει το 'ol', δηλαδή 'Golf' και 'Corrolla'.

```
SELECT * FROM shop
WHERE articlename like "%ol%";
```

Για τον προσδιορισμό του μήνα πωλήσεων μπορούμε να γράψουμε

```
SELECT article, articlename, datesold, MONTH(DateSold)
FROM shop;
```

Για να προσδιορίσουμε τις πωλήσεις που έγιναν τον 3<sup>ο</sup> Μήνα γράφουμε

```
SELECT article, articlename, datesold
FROM shop
WHERE MONTH(DateSold) =3;
```

Για να προσδιορίσουμε πόσες μέρες πέρασαν από την ημερομηνία πώλησης κάθε αυτοκινήτου, παίρνουμε την διαφορά μεταξύ της ημερομηνίας πώλησης και της σημερινής ημερομηνίας, αφού μετατρέψουμε τις δύο ημερομηνίες σε ημέρες.

```
SELECT articlename, datesold, (TO_DAYS(NOW())-TO_DAYS(datesold))
FROM shop;
```

Και για να έχουμε το αποτέλεσμα σε χρόνια, διαιρούμε με 365

```
SELECT articlename, datesold, (TO_DAYS(NOW())-TO_DAYS(datesold)) / 365
FROM shop;
```

## AGGREGATE FUNCTIONS: COUNT, SUM, MAX, MIN, AVG

Για το άθροισμα

```
SELECT SUM(price) FROM shop;
```

Για τη μέση τιμή

```
SELECT AVG(price) FROM shop;
```

Ή ακόμη για τη μέση τιμή των αυτοκινήτων μόνο του πωλητή A

```
SELECT AVG(price) FROM shop WHERE salesman='A';
```

Μέγιστη-Ελάχιστη τιμή

```
SELECT MIN(price) FROM shop;
```

```
SELECT MAX(price) FROM shop;
```

Για να δούμε ανεξαιρέτως το πλήθος όλων των εγγραφών

```
SELECT COUNT(*) from shop;
```

Με τη συνάρτηση count παίρνουμε το πλήθος των εγγραφών που ικανοποιούν συγκεκριμένα κριτήρια.

```
SELECT COUNT(*) FROM shop WHERE salesman='A';
```

Ακόμη για να δούμε πόσοι διαφορετικοί πωλητές υπάρχουν

```
SELECT COUNT(DISTINCT salesman) FROM shop;
```

Η χρήση aggregate function δεν μπορεί να συνδυαστεί με άλλες πληροφορίες ταυτόχρονα.

Η ακόλουθη αναζήτηση οδηγεί σε λάθος, δηλαδή δεν πραγματοποιείται

```
SELECT salesman, AVG(price) FROM shop;
```

Το πρόβλημα λύνεται με τη χρήση του group by

```
SELECT salesman, avg(price)
FROM shop
GROUP BY salesman;
```

Για κάθε πωλητή υπολογίζεται η μέση τιμή των αυτοκινήτων που πούλησε.

Μπορούμε να χρησιμοποιήσουμε και το order by ταυτόχρονα

```
SELECT salesman, avg(price)
FROM shop
GROUP BY salesman
ORDER BY price;
```

Ακόμη μπορούμε να έχουμε

```
SELECT salesman, COUNT(*) FROM shop GROUP BY salesman;
```

Πολλές φορές το αποτέλεσμα μιας αναζήτησης μπορεί να είναι ο συνδυασμός περισσότερων των μια ερωτήσεων. Για να βρούμε τον αριθμό, τον πωλητή και την τιμή του πιο ακριβού αυτοκινήτου, εκτελούμε τα ακόλουθα ερωτήματα

```
SELECT MAX(article) AS article FROM shop;
```

Και το αποτέλεσμα είναι έστω 19.95 και κατόπιν

```
SELECT article, salesman, price
FROM shop
WHERE price=19.95;
```

Πρέπει να τονιστεί ότι το group by πρέπει να χρησιμοποιηθεί απαραίτητα όταν επιλέγουμε στο ερώτημα να υπάρχουν aggregate functions και πεδία ταυτόχρονα.

## ΣΥΝΔΥΑΣΜΟΣ ΠΙΝΑΚΩΝ – JOINING TABLES

Ας υποθέσουμε ότι δημιουργούμε και έναν δεύτερο πίνακα events

```
create table events (
 service_id int(2) NOT NULL,
 article int(4) NOT NULL,
 technician varchar(20),
 type varchar(20),
 date DATE,
 PRIMARY KEY (service_id));
```

όπου καταγράφουμε για κάθε αυτοκίνητο τις πιθανές επισκευές που έχουν γίνει. Τα δεδομένα του πίνακα είναι τα εξής:

service_id	article	technician	type	date
1	1	George	service	2001-10-10
2	2	George	tire repair	2001-03-14
3	3	John	tire repair	2001-12-12
4	3	Jack	service	2000-05-08
5	3	Jack	oil change	2001-04-04
6	4	John	oil change	2001-12-01
7	4	Jim	service	2001-11-01

Οι δύο πίνακες shop και events έχουν μια σχέση που ορίζεται με το κοινό πεδίο article το οποίο και αποτελεί τη «γέφυρα διασύνδεσης», το κοινό κλειδί μεταξύ των δύο πινάκων. Με τη χρήση αυτού του κοινού πεδίου και την κατάλληλη σύνταξη της εντολής SELECT, μπορούμε να αντλήσουμε πληροφορίες και από τους δύο πίνακες ταυτόχρονα. Στο FROM αναγράφουμε και τους δύο πίνακες από όπου αντλούμε πληροφορίες. Στο WHERE εκτός από τις επιθυμητές συνθήκες πρέπει να αναφέρουμε και τη σχέση των δύο πινάκων. Επίσης για κάθε πεδίο πρέπει να αναφέρουμε και σε ποιόν πίνακα ανήκει.

Για να δούμε τις μάρκες των αυτοκινήτων που διορθώθηκαν από τον τεχνικό Jack, γράφουμε την ακόλουθη εντολή

```
SELECT shop.article,
 shop.articlename,
 events.technician
FROM shop, events
WHERE shop.article = events.article AND technician= "Jack";
```

Ένα αρκετά χρήσιμο χαρακτηριστικό της sql είναι η αντιστοίχιση ενός μικρού ονόματος στον πίνακα (alias a table). Το παραπάνω λοιπόν ερώτημα μπορεί να γραφεί και ως εξής:

```
SELECT A.article,
 A.articlename,
 B.technician
FROM shop A, events B
WHERE A.article = B.article AND technician= "Jack";
```

Επίσης για να δούμε ποια συνολικά αυτοκίνητα έκαναν αλλαγή λάστιχων και ποιες ημερομηνίες

```
SELECT shop.article,
 shop.articlename,
 shop.salesman,
 events.type
FROM shop, events
WHERE shop.article = events.article AND type="tire repair";
```

## Insert

Με την εντολή αυτή εισάγουμε data σε έναν πίνακα. Πρέπει να είμαστε απόλυτα προσεκτικοί στη σειρά και στους τύπους των πεδίων που εισάγουμε, ώστε να υπάρχει απόλυτη αντιστοιχία με τον ορισμό του πίνακα που έχει βεβαίως προηγηθεί.

Για παράδειγμα, για τον πίνακα shop που ορίσαμε παραπάνω, η εντολή INSERT μπορεί να γραφεί ως εξής και προσθέτει ένα νέο record στον πίνακα

```
INSERT INTO shop VALUES (1, 'audi', 'A', 3.45, '2001-1-1');
```

Για να εισάγουμε πολλές εγγραφές ταυτόχρονα μπορούμε να γράψουμε

```
INSERT INTO shop VALUES
(1, 'audi', 'A', 3.45, '2001-1-1'),
(2, 'Volkswagen Polo', 'B', 3.99, '2001-3-4'),
(3, 'Toyota Corolla', 'A', 10.99, '1999-12-10');
```

Για να εισάγουμε εγγραφές με Null τιμή σε κάποιο πεδίο, πρέπει να ορίσουμε τα πεδία εκείνα στα οποία εισάγουμε συγκεκριμένες τιμές.

```
INSERT INTO shop (article, price, datesold) VALUES ('30', 5.77, '2000-1-3');
```

## Update

Η γενική σύνταξη της εντολής είναι

```
UPDATE table_name
SET column_name=expression
WHERE condition
```

Ενημέρωση ενός συγκεκριμένου πεδίου μιας εγγραφής

```
UPDATE shop
SET salesman='D'
WHERE articlename='audi';
```

Ενημέρωση πολλών εγγραφών ταυτόχρονα

```
UPDATE shop
SET articlename='Volvo'
WHERE salesman='D';
```

Ενημέρωση πολλών πεδίων μιας εγγραφής

```
UPDATE shop
SET salesman='E', Price=5
WHERE articlename='Volkswagen Polo';
```

Ενημέρωση όλων των εγγραφών

```
UPDATE shop
SET price=price+50;
```

## Delete

Η χρήση της εντολής αυτής χρειάζεται μεγάλη προσοχή.

Η γενική σύνταξή της είναι

```
DELETE FROM table_name WHERE condition;
```

Διαγράφοντας ένα ή περισσότερα records που ικανοποιούν μια συνθήκη

```
DELETE from shop where salesman='D';
```

Διαγραφή όλων των εγγραφών από τον πίνακα

```
DELETE FROM shop;
```

Η διαφορά από την εντολή DROP TABLE είναι ότι με την delete παραμένει ο πίνακας σαν οντότητα στην βάση - απλώς είναι άδειος. Η DROP TABLE διαγράφει μαζί με τη δομή του πίνακα και τα δεδομένα.

## Ενώσεις (Joins)

Ο τυπικός τρόπος για ένωση δύο πινάκων είναι αυτός που είδαμε πριν:

```
SELECT companyname
FROM customers,orders
WHERE customerID=customerID;
```

Ένα τέτοιο ερώτημα θα επιστρέψει μόνο τα ονόματα πελατών που έχουν κάνει παραγγελίες. Ωστόσο υπάρχει και ένας 2<sup>ος</sup> τρόπος να συντάξουμε το join ερώτημα, που

κάνει την ίδια ακριβώς εργασία και επιστρέφει τα ίδια αποτελέσματα, γνωστό και ως ANSI-92 standard:

```
SELECT companyname
FROM customers
JOIN orders ON customer.customerID=orders.customerID;
```

Το γενικό συντακτικό του ANSI-92 standard είναι:

```
SELECT λίστα εμφανιζόμενων πεδίων
FROM όνομα πρώτου πίνακα
JOIN όνομα δεύτερου πίνακα ON κριτήριο ένωσης
WHERE συνθήκες;
```

Το συντακτικό που θα χρησιμοποιείται στη δόμηση ενός join ερωτήματος αφήνεται καθαρά στην προτίμηση του προγραμματιστή.

### **Cross/Cartesian Join**

Μια μορφή join που χρησιμοποιείται σπάνια και πολλές φορές κατά λάθος. Επίσης καταναλώνει περισσότερη υπολογιστική ισχύ και καλό είναι να αποφεύγεται.

Ουσιαστικά, επιστρέφει όλες τις εγγραφές από όλους τους πίνακες που συμπεριλαμβάνονται στο join:

```
select productid,productname from products,suppliers.
```

Σε μια τέτοια περίπτωση, θα επιστρέψει 2262 συνολικές εγγραφές, αν υποθεθεί ότι ο πίνακας products έχει 78 και ο πίνακας suppliers έχει 29 εγγραφές.

### **Inner Join**

Ένα inner join σημαίνει ότι όλες οι εγγραφές/γραμμές που είναι αταίριαστες αγνοούνται και εμφανίζονται μόνο οι κοινές από τους δύο πίνακες. Αυτός είναι ο προκαθορισμένος τύπος ένωσης 2 πινάκων, ενώ επίσης η χρήση της λέξης INNER είναι προαιρετική και μπορείτε να την παραλείψετε:

```
SELECT companyname
FROM customers
INNER JOIN orders ON customer.customerID=orders.customerID;
```

### **Left Join**

Ένα left join επιστρέφει όλες τις εγγραφές του αριστερού πίνακα, ασχέτως αν βρεθούν ταιριαστές εγγραφές στον δεξιό πίνακα:

```
SELECT companyname,orderID
FROM customers
LEFT JOIN orders ON customers.customerID=orders.customerID;
```

Στο παραπάνω παράδειγμα, όλες οι γραμμές / εγγραφές του πίνακα customers θα επιστραφούν ακόμη και αν δεν βρούν ταιριαστές εγγραφές σύμφωνα με την ένωση στο πίνακα orders.

## **Connecting in a mysql database using PHP**

DB connection:

Χρειαζόμαστε hostname, username και password.

```
$link_id=mysql_connect($hostname,$username,$password);
```

Ορίζουμε ποια ΒΔ (database) θα χρησιμοποιήσουμε:

```
mysql_select_db($dbname,$link_id);
```

Κλείσιμο της σύνδεσης με τη ΒΔ:

```
mysql_close($link_id)
```

## Basic mysql functions in PHP

Εκτέλεση ενός ερωτήματος (query):  
`$result_set=mysql_query($sql)`

Μεταφορά των δεδομένων από το συνολικό αποτέλεσμα (από το result set) σε μεταβλητές, προκειμένου να είναι εύκολη η διαχείριση τους

1<sup>ος</sup> τρόπος.

```
for ($i=0; $i<mysql_num_rows($result_set); $i++)
{
 $row=mysql_fetch_row($result);
 $var0=$row[0];
 $var1=$row[1];
}
```

2<sup>ος</sup> τρόπος.

```
while($row=mysql_fetch_row($result_set))
{
 $var0=$row[0];
 $var1=$row[1];
}
```

Αριθμός εγγραφών (rows) στο record set:

`mysql_num_rows($result_set)`

Αριθμός επηρεαζόμενων εγγραφών στο record set, ύστερα από μια εντολή INSERT, UPDATE OR DELETE:

`mysql_affected_rows($result_set)`

Διαφορά `mysql_fetch_row()` / `mysql_fetch_array()`:

`mysql_fetch_row()`: Κάθε στήλη επιστρέφεται ως array, ξεκινώντας από το 0

`mysql_fetch_array()`: μια διαφορετική, εκτεταμένη μορφή του [mysql\\_fetch\\_row\(\)](#). Μπορεί εκτός από indexes να χρησιμοποιεί και ονόματα στον πίνακα που αντιπροσωπεύει κάθε στήλη, ουσιαστικά δημιουργεί associative arrays με τα ονόματα των στηλών / πεδίων.

Απελευθέρωση μνήμης από το result set:

`mysql_free_result($result_set)`

## PHP and mysql tips

- Εναλλαγή χρωμάτων σε εμφάνιση πίνακα (alternative colors)

1ος τρόπος:

Χρήση του % (mod) operator, επιστρέφει 0 ή 1 για οποιαδήποτε τιμή της μεταβλητής.

`$i`: ο counter που λειτουργεί σαν A/A

`$color = ($i%2)?"000000#":"FFFFFF#";`

2ος τρόπος:

```
<?php
$bool = TRUE;
while ($array = mysql_fetch_array ($result))
{
 if ($bool == TRUE)
 {
 // print the white row
 $bool = FALSE
 }
 else
```

```
 {
 // print grey row
 $bool = TRUE
 }
 }
?>
```