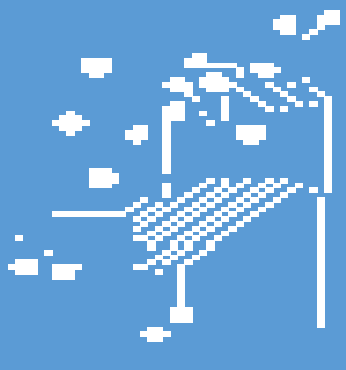


τμήμα μηχανικών πληροφορικής τ.ε.

ΑΝΑΠΤΥΞΗ ΕΚΠΑΙΔΕΥΤΙΚΟΥ ΠΕΡΙΕΧΟΜΕΝΟΥ ΓΙΑ ΤΗ ΓΛΩΣΣΑ PHP

ΒΑΣΙΛΟΠΟΥΛΟΣ ΠΑΝΑΓΙΩΤΗΣ 1336

ΠΑΠΑΚΩΣΤΟΠΟΥΛΟΥ ΜΑΡΙΑ 1785

ΕΠΙΒΛΕΠΩΝ: ΑΣΗΜΑΚΟΠΟΥΛΟΣ ΓΕΩΡΓΙΟΣ

ΑΝΤΙΡΡΙΟ 2017

Περιεχόμενα

1 Εισαγωγή

1.1 PHP και web εφαρμογές

Μια Web εφαρμογή στηρίζεται στην Client - Server αρχιτεκτονική. Ένας δικτυακός τόπος περιέχει ιστοσελίδες HTML και στοιχεία εκτελέσιμου κώδικα, τα οποία είναι αποθηκευμένα σε έναν Web server. Από την άλλη πλευρά υπάρχουν εργαλεία όπως οι φυλλομετρητές (Internet browser) στους υπολογιστές των χρηστών που συνδέονται με τον Web Server και γίνονται αποδέκτες του περιεχομένου των ιστοσελίδων. Ο χρήστης ζητάει το περιεχόμενο από τον Web Server στέλνοντας μια αίτηση σε μια συγκεκριμένη URL (Unified Reference Location), το περιεχόμενο αποστέλλεται και οι φυλλομετρητές αναλαμβάνουν να εμφανίσουν το περιεχόμενο στο χρήστη.

1.2 Αρχιτεκτονική Client-Server

Κατά τη διάρκεια των δύο τελευταίων δεκαετιών με την γεωμετρικής ταχύτητας ανάπτυξη του διαδικτύου και της υπηρεσίας του Παγκόσμιου Ιστού (WWW-World Wide Web) εμφανίστηκαν παράλληλα πολλές τεχνολογίες ανάπτυξης περιεχομένου του ιστοχώρου (Web Development Technologies). Θα μπορούσαμε τις τεχνολογίες αυτές να τις διακρίνουμε σε δύο κατηγορίες σε σχέση με το μοντέλο Client Server στο οποίο στηρίζεται όπως είπαμε το Internet.

1.3 Προγραμματισμός Εφαρμογής σε επίπεδο Client-Side

Οι τεχνολογίες που χρησιμοποιούνται για την ανάπτυξη ενός δικτυακού τόπου σε επίπεδο client είναι περιγράφονται αναλυτικά παρακάτω.

1.3.1 Γλώσσες Script

Οι γλώσσες script στην ανάπτυξη ιστοσελίδων, χρησιμοποιήθηκαν αρχικά ως κώδικας ο οποίος θα εκτελούνταν από τους servers. Αργότερα η ανάπτυξη browsers οι οποίοι θα μπορούσαν να αποκωδικοποιούν και να εκτελούν κώδικα και στη πλευρά του client έδωσαν μεγάλες δυνατότητες στους σχεδιαστές ιστοσελίδων.

1.3.2 HTML 4.0 - HyperText Markup Language

Η γλώσσα HyperText Markup Language (HTML) είναι η βασική γλώσσα που χρησιμοποιείται στον Παγκόσμιο Ιστό για την περιγραφή της δομής και της μορφής του περιεχομένου ενός εγγράφου. Οι φυλλομετρητές (browsers) μεταφράζουν τη γλώσσα αυτή έτσι ώστε να παρουσιάσουν στο χρήστη το περιεχόμενο του εγγράφου με τον τρόπο αναπαράστασης που περιγράφεται από τη γλώσσα.

Έτσι browsers οι οποίοι εμφανίζουν το περιεχόμενο στην οθόνη, το κείμενο που παρεμβάλλεται ανάμεσα στ tags … το εμφανίζουν με έντονη γραφή ενώ τα προγράμματα ανάγνωσης αυτών των εγγράφων διαβάζουν με μεγαλύτερη έμφαση το κείμενο αυτό. Σε συνδυασμό με τα Cascading Style Sheets (CSS) ο συντάκτης ενός τέτοιου εγγράφου μπορεί να προσδιορίσει πως τα στοιχεία του εγγράφου θα εμφανιστούν παρακάμπτοντας τις προεπιλογές ενός browser.

Οι βασικές αρχές της HTML στην έκδοση 4.0 είναι οι εξής:

- Διαχωρισμός του περιεχομένου από τον τρόπο παρουσίασης μέσω των style sheets. Αποτελεί τη βασική αρχή σχεδίασης Web περιεχομένου. Η HTML 4.0 κάνει σαφή διαχωρισμό της δομή από τον τρόπο παρουσίασης του περιεχομένου για την καλύτερη και αποδοτικότερη δημιουργία ιστοσελίδων. Αυτό επιτυγχάνεται με τη χρήση των cascade style sheets (CSS).
- Προσβασιμότητα και Διεθνής Προτυποποίηση. Στην HTML 4.0 βασική αρχή θεωρείται η πρόσβαση στο περιεχόμενο για άτομα που χρησιμοποιούν ειδικούς browsers είτε λόγω μειωμένων ικανοτήτων ή λόγω έλλειψης τηλεπικοινωνιακής υποδομής. Επίσης είναι σημαντική η υποστήριξη κωδικοποιήσεων για όλες τις γλώσσες.
- Αποδοτικότερη μετάφραση των εγγράφων Web. Στην HTML 4.0 προστέθηκα αρκετά στοιχεία για την καλύτερη και αποδοτικότερη μετάφραση των εντολών περιγραφής του περιεχομένου.
- Καθορισμός τριών και μόνο DTD. Στην HTML 4.0 υπάρχουν τρεις τύποι εγγράφων που μπορούν να χρησιμοποιηθούν ως Document Type Definitions (DTD): Strict, Transitional, και Frameset. Στο τύπο Strict η μορφοποίηση του περιεχομένου μιας ιστοσελίδας βασίζεται μόνο στο αντίστοιχο CSS αγνοώντας τα χαρακτηριστικά περιγραφής στοιχείων που βρίσκονται μέσα στο HTML έγγραφο. Στο Transitional DTD γίνεται μια παραχώρηση και λαμβάνονται υπόψη κάποια χαρακτηριστικά που περιγράφουν τη μορφοποίηση μέσα στο έγγραφο. Τέλος ο τύπος Frameset DTD καθορίζει τον τρόπο χρήσης των frames στην HTML 4.0.

Με την έκδοση 4.01, δόθηκαν λύσεις σε μερικά προβλήματα της HTML 4.0.

1.4 Server-Side

Πρόκειται για τεχνολογίες προγραμματισμού και ανάπτυξης εφαρμογών οι οποίες εκτελούνται στην μεριά του web server πριν το περιεχόμενο αποσταλεί στον web browser του τελικού χρήστη.

- PHP. Η PHP είναι μια διαδεδομένη γλώσσα script που εκτελείται σε επίπεδο server και χρησιμοποιείται στη δημιουργία ιστοτόπων δυναμικού περιεχομένου. Είναι γλώσσα ανοικτού κώδικα.

- MySQL. Η MySQL είναι ένα ισχυρό Σύστημα Διαχείρισης Βάσεων Δεδομένων. Πρόκειται για λογισμικό ανοικτού κώδικα και η διαχείριση των βάσεων γίνεται με τη γλώσσα Structured Query Language (SQL). Συνδυάζεται συνήθως με PHP και τη υποστηρίζεται από όλα τα λειτουργικά συστήματα. Βασικός ανταγωνιστής είναι ο SQL Server της Microsoft με αρκετά υψηλό κόστος εγκατάστασης και λειτουργίας.

1.5 Προγραμματισμός σε επίπεδο Server

Πιο αναλυτικά οι πιο διαδεδομένες τεχνολογίες ανάπτυξης δυναμικών εφαρμογών σε επίπεδο Server είναι οι παρακάτω.

1.6 PHP

Οι σελίδες του Παγκόσμιου Ιστού περιέχουν κώδικα script σε γλώσσα HTML (Hyper Text Markup Language). Σε μια Web εφαρμογή όμως εκτός των περιεχομένων, οι ιστοσελίδες (server pages) περιέχουν ενσωματωμένο και εκτελέσιμο κώδικα ο οποίος εκτελείται στον Εξυπηρετητή χωρίς να είναι ορατός στον τελικό χρήστη. Τέτοιες σελίδες είναι οι PHP σελίδες (Hypertext Preprocessor) οι οποίες και περιέχουν ενσωματωμένο κώδικα PHP. Η PHP είναι μια γλώσσα script από την πλευρά του Εξυπηρετητή , σχεδιασμένη ειδικά για το Web. Ξεκίνησε αρχικά σαν μια σύντομη έκδοση της Perl από τον Rasmus Lerdorf το 1994. Δανείστηκε στοιχεία από τη C, τη Java και την Perl και αναπτύχθηκε έτσι ώστε να μπορεί να ενσωματωθεί σε αρχεία HTML με επέκταση ".php", ".php3", ή ".phtml". Μια σελίδα PHP περνά από επεξεργασία από ένα συμβατό διακομιστή του Παγκόσμιου Ιστού (π.χ. Apache), ώστε να παραχθεί σε πραγματικό χρόνο το τελικό περιεχόμενο, που θα σταλεί στο πρόγραμμα περιήγησης των επισκεπτών σε μορφή κώδικα HTML

Βασικό της χαρακτηριστικό είναι ότι οι σελίδες αυτές σχεδιάζονται δυναμικά ανάλογα με την εκτέλεση του κώδικα. Τα βασικά πλεονεκτήματα των PHP σελίδων είναι τα εξής:

- Είναι πολύ εύκολη στην εκμάθηση της. Ο προγραμματισμός σε PHP είναι οικείος σε προγραμματιστές C, Perl και Java
- Υποστηρίζει πολλές πλατφόρμες (Windows, Linux, Unix, κα)
- Υπάρχει συμβατότητα με σχεδόν όλους τους εξυπηρετητής (server)s (Apache, IIS, κα)
- Παρέχει εύκολη συνδεσιμότητα με Βάσεις Δεδομένων όπως MySQL, Oracle, Sybase, PostgreSQL, Generic ODBC κα.
- Παρέχει ενσωματωμένες βιβλιοθήκες για πολλές Web διεργασίες όπως να δημιουργήσει δυναμικά αρχεία εικόνων, να αναλύσει XML, να στείλει ηλεκτρονικό μήνυμα, να δημιουργήσει cookies και PDF έγγραφα
- Ανήκει στην κατηγορία του Λογισμικού Ανοικτού Κώδικα (Open Source software – OSS).
- Συνεργάζεται με την επίσης Ανοικτού Κώδικα βάση Δεδομένων MySQL.
- Είναι διαθέσιμη δωρεάν http://www.php.net. Η χρήση είναι δωρεάν.

1.7 Βάση Δεδομένων MySQL

Η web εφαρμογή χρησιμοποιεί συνήθως μια βάση δεδομένων για την καταχώρηση και ανάκτηση δεδομένων. Στην αρχιτεκτονική πελάτης - εξυπηρετητής υπάρχει ένα σύστημα Διαχείρισης Βάσης Δεδομένων συνήθως Σχεσιακής (Relational Database System - RDBMS) όπου καταχωρούνται τα δεδομένα. Ανάλογα με τις ενέργειες και τις αιτήσεις του χρήστη, ο εξυπηρετητής (server) επικοινωνεί με το σύστημα διαχείρισης της βάσης δεδομένων και εκτελεί ερωτήματα (queries) σε γλώσσα SQL.

Το σύστημα διαχείρισης της Βάσης Δεδομένων με τη σειρά του απαντάει σε αυτά τα queries του εξυπηρετητή είτε αποστέλλοντας τα δεδομένα που προέκυψαν σαν αποτελέσματα των ερωτημάτων ή εκτελώντας κάποια εισαγωγή ή διαγραφή δεδομένων στην περίπτωση εντολών insert ή update ή delete. Η επικοινωνία μεταξύ εφαρμογής και Βάσης Δεδομένων γίνεται με τη χρήση οδηγών (Database Connectivity drivers).

Η MySQL είναι ένα Σύστημα Διαχείρισης Σχεσιακής Βάσης Δεδομένων και περιέχει και έναν μικρό server της βάσης. Είναι ένα πολύ γρήγορο δυνατό, *σύστημα*

διαχείρισης σχεσιακών βάσεων δεδομένων. Ο διακομιστής MySQL ελέγχει την πρόσβαση στα δεδομένα για να διασφαλίσει ότι πολλοί χρήστες μπορούν να δουλεύουν ταυτόχρονα, για να παρέχει γρήγορη πρόσβαση και για να διασφαλίσει ότι μόνο εξουσιοδοτημένοι χρήστες μπορούν να έχουν πρόσβαση. Χρησιμοποιεί την SQL (Structured Query Language) γλώσσα ερωτημάτων για βάσεις δεδομένων. Η MySQL είναι διαθέσιμη από το 1996, αλλά η ιστορία της ξεκινάει από το 1979. Αναπτύχθηκε σαν μια εφαρμογή της γλώσσας SQL από την TcX. Είναι αρκετά σταθερό σύστημα και πολύ ευέλικτο. Υποστηρίζει όλες τις λειτουργίες και τους τύπους δεδομένων της standard. Είναι πλέον παγκοσμίως η πιο δημοφιλής βάση δεδομένων ανοιχτού κώδικα και έχει κερδίσει αρκετές φορές το βραβείο Choice Award του Linux Journal Readers. Εκτός των παραπάνω τα πιο σημαντικά χαρακτηριστικά της MySQL είναι τα ακόλουθα:

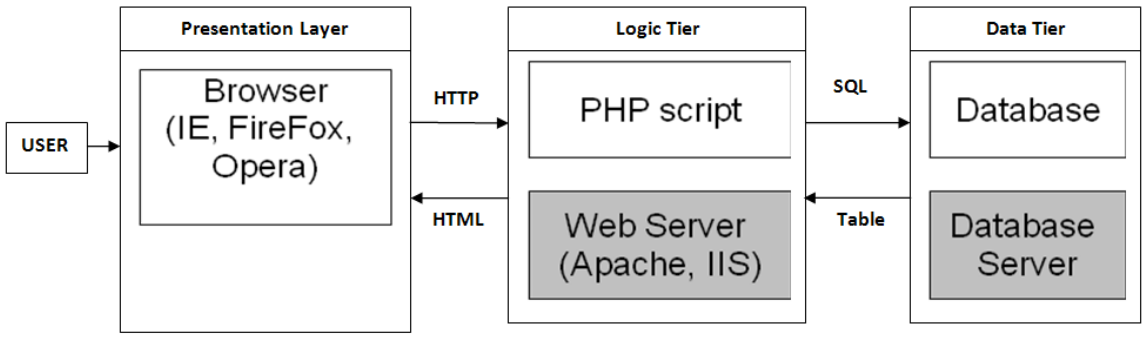
- Η MySQL ανήκει στο λογισμικό Ανοικτού Κώδικα (Open Source). Η χρήση της είναι δωρεάν
- Είναι γρήγορη και υποστηρίζει multi-thread και πολυχρηστικό περιβάλλον.
- Είναι συμβατή με πολλές και ποικίλες πλατφόρμες.

1.8 PHP – MySQL

Η χρήση PHP με το σύστημα διαχείρισης της MySQL έχει σαν βασικό πλεονέκτημα τη συμβατότητα με πολλές πλατφόρμες και επίσης ότι ανήκουν και οι δύο στις εφαρμογές Ανοικτού Κώδικα και τα δικαιώματα χρήσης τους είναι δωρεάν. Εξαιτίας αυτών των χαρακτηριστικών, οι διακομιστές που φιλοξενούν δυναμικές σελίδες PHP συνδυάζουν υπηρεσίες γι αυτές τις δύο πλατφόρμες και την άμεση διασύνδεσή τους. Η λειτουργία τους βασίζεται στον εκτελέσιμο κώδικα που είναι ενσωματωμένος στις PHP σελίδες. Ο κώδικας αυτός εκτελείται στον διακομιστή. Ο κώδικας εκτελεί ερωτήματα σε SQL τα οποία μεταβιβάζονται μέσω ενός middleware της MySQL στη βάση MySQL. Ανάλογα με την αίτηση του συστήματος προς τη βάση εκτελείται είτε μια εντολή ανάκτησης, ή εγγραφής ή μια τροποποίησης ή διαγραφής δεδομένων στη Βάση. Η αίτηση μεταβιβάζεται στη Βάση Δεδομένων και το σύστημα διαχείρισης δεδομένων επιστρέφει μια απάντηση στο Web server με τα αποτελέσματα. Στη συνέχεια τα δεδομένα χρησιμοποιούνται από τον εκτελέσιμο κώδικα στη δημιουργία του HTML περιεχομένου της δυναμικής ιστοσελίδας που τελικά αποστέλλεται στην πελάτης (client) εφαρμογή δηλ. τον web browser του χρήστη όπου και προβάλλεται το περιεχόμενο που ζήτησε.

Αρχιτεκτονική 3-Tier

Το μοντέλο Πελάτης - Εξυπηρετητής (client - server) αναφέρεται και ως αρχιτεκτονική 2-tier διότι αποτελείται από δύο μέρη. Στις περιπτώσεις που τα δεδομένα αποθηκεύονται σε κάποια βάση δεδομένων σε έναν ξεχωριστό εξυπηρετητή, το μοντέλο αποτελείται από τρία διαφορετικά συστατικά. Οι εφαρμογές Πελάτης στέλνουν δεδομένα στην εφαρμογή Εξυπηρετητή η οποία αποθηκεύει τα δεδομένα σε ένα ξεχωριστό Σύστημα Διαχείρισης Βάσεων. Το ξεχωριστό αυτό σύστημα είναι υπεύθυνο για τη διαχείριση και τη διανομή των δεδομένων. Συνεπώς το συγκεκριμένο μοντέλο αποτελείται από τρία συστατικά και ονομάζεται αρχιτεκτονική 3-tier. Η προσέγγιση αυτή είναι καλύτερη από το απλό 2-tier μοντέλο επειδή απομονώνει τη λειτουργία της αποθήκευσης και διαχείρισης των δεδομένων από τη λειτουργία της επεξεργασίας τους με συνέπεια να απελευθερώνονται σημαντικοί πόροι για τον εξυπηρετητή επεξεργασίας των δεδομένων. Επιπλέον το μοντέλο 3-tier αυξάνει την συνδεσιμότητα και επεκτασιμότητα όλου του πληροφοριακού συστήματος αφού η διαχείριση των δεδομένων αποδεσμεύεται από τη συγκεκριμένη εφαρμογή και εκτελείται από ένα σύστημα που μπορεί να είναι συμβατό και διασυνδεδεμένο και με άλλες εφαρμογές [Edelstein 1994]. Η διαδικασία παρουσιάζεται στο Σχήμα 1.



2 Βασικοί συντακτικοί κανόνες της PHP

Η PHP χρησιμοποιεί τα σύμβολα <?php και ?> για να εισαγάγει ένα μπλοκ σκριπτ μέσα σε ένα έγγραφο html/xhtml. Εναλλακτικά, μπορεί να χρησιμοποιείται και τα πιο σύντομα σύμβολα <? και ?> ή ακόμα και τα σύμβολα της ASP τα οποία είναι <% και %> αρκεί να υπάρχει η αντίστοιχη ρύθμιση στο server. Ωστόσο, για μέγιστη συμβατότητα είναι καλό να μπαίνει ο πρώτος συμβολισμός.

```
<?php

?>
```

Ένα αρχείο php κανονικά περιέχει κώδικα html και μπλοκ από κώδικα σκριπτ php. Το παρακάτω παράδειγμα είναι ένα απλό αρχείο php το οποίο περιέχει ένα σκριπτ php και το οποίο στέλνει στο πρόγραμμα περιήγησης το κείμενο " hello world ".

```
<?php
echo("hello world");
?>
```

Όπως αναφέρθηκε, ένα έγγραφο μπορεί να περιέχει περισσότερα από ένα μπλοκ. Δείτε το παρακάτω παράδειγμα.

```
<?php
echo("hello ");
?>

<br />

<?php
echo("world");
?>
```

Το αποτέλεσμα θα είναι:

hello
world

Εκτός από την εντολή echo μπορείτε να χρησιμοποιήσετε και την εντολή print για να αποστείλετε κείμενα στο πρόγραμμα περιήγησης.

- Στην php, κάθε εντολή πρέπει να τερματίζεται με το ελληνικό ερωτηματικό (;).
- Η php είναι μια γλώσσα case sensitive. Αυτό σημαίνει ότι κάνει διάκριση ανάμεσα σε πεζά και κεφαλαία γράμματα. Άρα το echo δεν είναι το ίδιο με το Echo.
- Οι εντολές της php γράφονται κατά κανόνα με πεζά γράμματα.

2.1 Οι μεταβλητές στην PHP

2.1.1 Ονόματα μεταβλητών

Βασικός κανόνας στην php είναι ότι τα ονόματα των μεταβλητών ξεκινούν πάντα με τον χαρακτήρα $. Έτσι, η php μπορεί να αναγνωρίσει τις μεταβλητές ανάμεσα σε άλλα κείμενα. Το μήκος του ονόματος μπορεί να είναι από έναν χαρακτήρα μέχρι κάποιο μεγάλο όνομα που να είναι αρκετά περιγραφικό.

- Κάθε όνομα μεταβλητής πρέπει να αρχίζει από ένα λατινικό γράμμα ή την κάτω παύλα "_" ενώ δεν μπορεί να ξεκινάει από αριθμό.
- Δεν επιτρέπονται μέσα στο όνομα χαρακτήρες άλλοι εκτός από αλφαριθμητικοί και η κάτω παύλα (A-z, 0-9).
- Απαγορεύονται επίσης τα διαστήματα στα ονόματα των μεταβλητών.
- Τέλος τα ονόματα είναι διαφορετικά αν γράφονται με κεφαλαία ή πεζά. ($a και $A είναι δύο διαφορετικές μεταβλητές)

2.1.2 Εμβέλεια μεταβλητών (scope)

Ανάλογα με την προσβασιμότητα που μπορεί να έχουμε, οι μεταβλητές χωρίζονται σε:

- local
- global
- static

Οι local μεταβλητές είναι αυτές που είναι δηλωμένες μέσα σε μια συνάρτηση. Είναι προσβάσιμες μέσα από την συνάρτηση και ζουν όσο ζει και η συνάρτηση.
Οι global μεταβλητές είναι αυτές που είναι δηλωμένες έξω από συναρτήσεις. Είναι προσβάσιμες από όλες τις συναρτήσεις και ζουν όσο ζει το μπλοκ του σκριπτ.

2.1.3 Στατικές μεταβλητές

Οι μεταβλητές static είναι αυτές που παρόλο που είναι είναι δηλωμένες μέσα σε μια συνάρτηση είναι προσβάσιμες μέσα από την συνάρτηση και διατηρούν την ισχύ τους ακόμα και όταν τελειώσει η συνάρτηση. Έτσι όταν ξανακληθεί η ίδια συνάρτηση η static θα διατηρεί την τιμή της. Συνήθως όταν μια συνάρτηση τερματίζεται, όλες οι μεταβλητές της διαγράφονται. Στην περίπτωση που ο προγραμματιστής επιθυμεί οι τοπικές μεταβλητές να μη διαγραφούν τις χαρακτηρίζει ως static.

Στη συνέχεια δίνεται ένα παράδειγμα συνάρτησης που περιέχει μια τέτοια μεταβλητή.

Κώδικας	Αποτέλεσμα

<?php
function myTest() {
    static $x = 0;
    echo $x;
    $x++;
}
 	
myTest();
myTest();
myTest();
?>
	0
	
	1
	
	2

2.1.4 Τύποι μεταβλητών

Η PHP μετατρέπει αυτόματα τις μεταβλητές στον κατάλληλο τύπο ανάλογα με την τιμή που αυτή λαμβάνει αντίθετα με άλλες παρόμοιες συντακτικά γλώσσες όπως η C και η Java όπου ο προγραμματιστής πρέπει να δηλώσει εκ των προτέρων τον τύπο πριν η μεταβλητή λάβει τιμή.

Οι τύποι μεταβλητών που υποστηρίζονται είναι οι ακόλουθοι:

2.1.5 Ακέραιος

Ο ακέραιος τύπος δεδομένων υποστηρίζει μη δεκαδικούς αριθμούς οι οποίοι κυμαίνονται από -2,147,483,648 μέχρι 2,147,483,647.
Οι κανόνες που πρέπει να πληρούν είναι οι εξής:

- Ένας ακέραιος πρέπει να έχει τουλάχιστον 1 ψηφίο και δεν πρέπει να έχει υποδιαστολή.
- Μπορεί να είναι είτε αρνητικός ή θετικός.
- Οι ακέραιοι μπορούν να προσδιοριστούν με 3 τρόπους: ως δεκαδικοί (με βάση το 10), δεκαεξαδικοί (με βάση το 16 και πρόθεμα το 0x) ή οκταδικοί (με βάση το 8 και πρόθεμα το 0)

Στο ακόλουθο παράδειγμα ο $x είναι ακέραιος όπως επιστρέφει η συνάρτηση var_dump() της PHP.

```
<?php
$x = 5985;
var_dump($x);
?>
```

2.1.6 Πραγματικοί αριθμοί κινητής υποδιαστολής

Ένας αριθμός κινητής υποδιαστολής είναι ένας αριθμός με υποσδιαστολή ή εκφρασμένος σε εκθετική μορφή. Οι αριθμοί αυτοί είναι τύπου float. Στο παράδειγμα που ακολουθεί ο $x είναι τύπου float. Η συνάρτηση var_dump() επιστρέφει τον τύπο και την τιμή της x:

```
<?php
$x = 10.365;
var_dump($x);
?>
```

2.1.7 Λογικοί (Boolean)

Οι λογικές μεταβλητές είναι εκείνες που μπορούν να λάβουν την λογική τιμή TRUE ή FALSE.

```
$x = true;
$y = false;
```

2.2 Πίνακες (Array)

Ο τύπος δεδομένων πίνακα (array) έχει τη δυνατότητα να αποθηκεύει πολλά δεδομένα κάτω από ένα κοινό όνομα.

```
<?php
$cars = array("Volvo","BMW","Toyota");
var_dump($cars);
?>
```

2.3 Αντικείμενο Object

Ένα αντικείμενο είναι ένας σύνθετος τύπος δεδομένων ο οποίος αποθηκεύει πολλά δεδομένα καθώς και πληροφορίες επεξεργασίας των δεδομένων αυτών. Όπως περιγράφεται στο τμήμα του Αντικειμενοστραφούς προγραμματισμού σε PHP ένα αντικείμενο δηλώνεται ειδικά και κατασκευάζεται με βάση μια περιγραφή που ονομάζεται κλάση. Μια κλάση είναι μια δομή που περιγράφει ιδιότητες και μεθόδους σχετικές με κάποια δεδομένα.

```
<?php
class Car {
   function Car() {
      $this->model = "VW";
   }
```

```
}

// create an object
$herbie = new Car();

// show object properties
echo $herbie->model;
?>
```

2.4 Τιμή NULL

Ο τύπος Null είναι ένας ειδικός τύπος δεδομένων που μπορεί να έχει μόνο μια τιμή, τη NULL.

Μια μεταβλητή NULL είναι μια μεταβλητή η οποία δεν έχει κάποια τιμή. Όταν μια μεταβλητή δημιουργείται και δεν της αποδίδεται καμία τιμή τότε προεπιλεγμένα παίρνει την τιμή NULL. Επίσης ο τύπος Null χρησιμοποιείται για να αδειάσει το περιεχόμενο μιας μεταβλητής.

```
<?php
$x = "Hello world!";
$x = null;
var_dump($x);
?>
```

2.5 Σταθερές

Μια σταθερά είναι ένας προσδιοριστής (ένα συμβολικό όνομα) για μια τιμή. Η τιμή της σταθεράς δεν μπορεί να αλλάξει κατά τη διάρκεια εκτέλεσης του κώδικα. Το όνομα μιας σταθεράς θα πρέπει να ξεκινά με ένα λατινικό γράμμα ή την κάτω παύλα χωρίς τη χρήση του προθέματος $. Αντίθετα με τις μεταβλητές, οι σταθερές έχουν καθολική εμβέλεια σε όλον τον κώδικα.

Μια σταθερά δημιουργείται με τη χρήση της συνάρτησης define(), η οποία συντάσσεται ως εξής:

define(*όνομα*, *τιμή*, αν θα είναι *case-insensitive*)

Παράμετροι:

- όνομα: Προσδιορίζει το όνομα της σταθεράς
- value: Προσδιορίζει την τιμή της σταθεράς
- case-insensitive: Προσδιορίζει με true ή false αν το όνομα της σταθεράς θα είναι case-insensitive. Προεπιλεγμένα είναι false.

Στο παρακάτω παράδειγμα φαίνεται ο ορισμός μιας case-insensitive σταθεράς

```
<?php
define("GREETING", "Welcome to W3Schools.com!", true);
```

```
echo greeting;
?>
```

2.6 Τελεστές Πράξεων

Οι τελεστές χρησιμοποιούνται σε συνδυασμό με μία ή περισσότερες τιμές δεδομένων για να δώσουν ένα συγκεκριμένο αποτέλεσμα. Οι τελεστές διακρίνονται σε κατηγορίες οι οποίες είναι:

Αριθμητικοί τελεστές

Τελεστής	**Περιγραφή**
+	Πρόσθεση
-	Αφαίρεση
*	Πολλαπλασιασμός
/	Διαίρεση
%	Υπόλοιπο διαίρεσης (Modulus)
++	Αύξηση κατά ένα
--	Μείωση κατά ένα

Τελεστές ορισμού τιμών

Τελεστής	**Παράδειγμα**	**Αντί για**
=	x=y	
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
%=	x%=y	x=x%y

Συγκριτικοί τελεστές

Τελεστής	**Περιγραφή**
==	ίσον
!=	όχι ίσον
>	μεγαλύτερο
<	μικρότερο
>=	μεγαλύτερο ή ίσον
<=	μικρότερο ή ίσον

Παραδείγματα

1==2 (είναι το 1 ίσο με 2;) είναι false

1!=2 (είναι το 1 διάφορο από 2;) είναι true

1>2 (είναι το 1 μεγαλύτερο του 2;) είναι false

1<=2 (είναι το 1 μικρότερο ή ίσο από 2;) είναι true

Λογικοί τελεστές

Τελεστής	**Περιγραφή**
&&	Λογική Σύζευξη
||	Λογική Διάζευξη
!	Λογική Άρνηση

Παράδειγμα	**Αποτέλεσμα**
x=6; y=3; (x < 10 && y > 1)	true
x=6; y=3; (x==5 || y==5)	false
x=6; y=3; !(x==y)	true

3 Αλγοριθμικές Δομές

Οι προγραμματισμός σε PHP βασίζεται στις αρχές του δομημένου προγραμματισμού οι οποίες και χρησιμοποιούνται είτε στον διαδικασιακό προγραμματισμό (procedural programming) ή στον αντικειμενοστραφή (object – oriented). Και οι δύο αυτές μεθοδολογίες υποστηρίζονται από την PHP και χρησιμοποιούν σαν βασικά δομικά στοιχεία τις βασικές αλγοριθμικές δομές δηλαδή τη δομή ακολουθίας (σειριακή δομή), δομή επιλογής (δομή ελέγχου) και επανάληψης.

3.1 Δομή Επιλογής

Οι Δομές Επιλογής αποτελούν βασικές αλγοριθμικές δομές, υλοποιούνται στην PHP και χρησιμοποιούνται για να εκτελεστεί ή να αγνοηθεί ένα συγκεκριμένο τμήμα κώδικα όταν ικανοποιείται ή δεν ικανοποιείται μια συνθήκη.
Δομή απλής επιλογής με την εντολή if
Η δομή της εντολής if στην PHP είναι παρόμοια με τις γλώσσες Java και C και η σύνταξή της είναι η παρακάτω:

```
if(συνθήκη)
{
 ...
}
```

Εάν η συνθήκη ικανοποιείται δηλαδή το αποτέλεσμα της έκφρασης είναι αληθής (true) τότε και μόνο τότε το τμήμα κώδικα που εμφωλεύεται στη δομή (ανάμεσα στα άγκιστρα) θα εκτελεστεί.

```
<?php
 if(συνθήκη)
 {
  ...
 }
?>
```

Παράδειγμα

Κώδικας	Αποτέλεσμα

<?php
$i=-1;
if($i<0)
{
	0

<?php $i=0; } echo $i; ?>	

3.2 Δομή σύνθετης επιλογής με την εντολή if...else

Η δομή if...else στην PHP είναι επίσης παρόμοια με τις γλώσσες Java και C . Πρόκειται για ομάδα εντολών όπου θα εκτελεστεί είτε το τμήμα κώδικα που είναι εμφωλευμένο στο if ή το τμήμα κώδικα που είναι εμφωλευμένο στο else υποχρεωτικά. Αν η συνθήκη είναι αληθής (true) τότε θα εκτελεστεί το εμφωλευμένο τμήμα κώδικα που αντιστοιχεί στο if, σε κάθε άλλη περίπτωση θα εκτελεστούν οι εντολές που αντιστοιχούν στο else.

Η σύνταξη της δομής if...else είναι η παρακάτω:

```
<?php
if(συνθήκη)
{
 ...
}
else
{
 ...
}
?>
```

Παράδειγμα

Κώδικας	Αποτέλεσμα

<?php
$d=7;
if ($d%2==0)
{
echo "Ζυγός αριθμός";
}
else
{
echo "Μονός αριθμός ";
}
	Μονός αριθμός

?>	

Στην περίπτωση που το τμήμα εμφωλευμένων εντολών αποτελείται από μια μόνο εντολή, τότε τα άγκιστρα παραλείπονται.

Παράδειγμα

Κώδικας	Αποτέλεσμα

<?php
$d=7;
if ($d%2==0)
echo "Ζυγός αριθμός";
else
echo "Μονός αριθμός ";
 	
?>
	Μονός αριθμός

3.2.1 Δομή πολλαπλής επιλογής με την εντολή elseif

Η δομή πολλαπλής επιλογής επίσης υποστηρίζεται στην PHP αντίστοιχα με ίδιο τρόπο σύνταξης όπως οι Java και C. Με την εντολή **elseif,** αν δεν ικανοποιείται η αρχική συνθήκη (if), εκτελείται ένα τμήμα κώδικα εφόσον ικανοποιείται η αντίστοιχη υπο-συνθήκη. Οι διαδοχικές ερωτήσεις γίνονται με την σειρά που έχουν διατυπωθεί Αν η δομή στο τέλος κλείνει με την εντολή else και δεν ικανοποιείται καμία από τις συνθήκες ή υπο-συνθήκες, τότε εκτελείται υποχρεωτικά η else.

Η σύνταξη της δομής elseif είναι η παρακάτω:

```
<?php
 if (συνθήκη)
 {
  ...
 }
 elseif (συνθήκη)
 {
  ...
 }
 elseif (συνθήκη)
 {
```

```
  ...
 }
  ...
 else
 {
  ...
 }
?>
```

Παράδειγμα

Κώδικας	Αποτέλεσμα για 18/2/2017

<?php
$d=date("D");
if ($d=="Sat")
{
  echo "Σαββατο";
}
elseif ($d=="Sun")
{
  echo "Κυριακή";
}
else
{
  echo "Καθημερινή";
}
?>
	Σαββατο

3.2.2 Δομή πολλαπλής επιλογής με την εντολή Switch

Η εντολή **switch** είναι ισοδύναμη της πολλαπλής δομής επιλογής σε κάποιες περιπτώσεις με διαφορετικό όμως τρόπο σύνταξης.

Η δομή της πρότασης **switch** είναι η παρακάτω:

```
switch (x)
{
 case value_1:
  Κώδικας προς εκτέλεση εάν x = value _1;
  break;
```

```
 case value_2:
  Κώδικας προς εκτέλεση εάν x = value _2;
  break;
 case value_3:
  Κώδικας προς εκτέλεση εάν x = value _3;
  break;
 default:
  Κώδικας προς εκτέλεση εάν η x δεν είναι ίση με καμία από τις
παραπάνω;
}
```

Παράδειγμα

Έστω ότι η μεταβλητή $x είναι ίση με 3. Όταν ακολουθήσει η πρόταση switch θα ελέγξει την τιμή της $x και θα μεταφερθεί στην περίπτωση 2 (case 2:) όπου και θα εκτελέσει τον κώδικα.

Στο τέλος κάθε case πρέπει να υπάρχει ένα break διότι διαφορετικά θα συνεχιστεί η εκτέλεση του κώδικα του επόμενου case.

Κώδικας	Αποτέλεσμα

<?php
$x=3;
switch ($x)
{
case 1:
echo "No 1";
break;
case 2:
echo "No 2";
break;
case 3:
echo "No 3";
break;
default:
echo "No number between 1 and 3";
}
?>
	No 3

3.3 Δομή επανάληψης

Η PHP υποστηρίζει τις βασικές δομές επανάληψης που είναι γνωστές από την Java και C και οι οποίες επιτρέπουν την επαναλαμβανόμενη εκτέλεση ενός τμήματος κώδικα όσο πληρούται κάποια συνθήκη. Επίσης με την εντολή foreach επεκτείνει τη χρήση των δομών επανάληψης και σε άλλες περιπτώσεις.

3.3.1 Η εντολή επανάληψης for

Η εντολή for χρησιμοποιείται για συγκεκριμένο πλήθος επαναλήψεων εκτέλεσης ενός τμήματος κώδικα. Το πλήθος αυτό είναι είτε ορισένο από τον προγραμματιστή ή από τον χρήστη.

Στην εντολή for χρησιμοποιείται συνήθως μια μεταβλητή σε ρόλο αριθμητής (counter) για τη μέτρηση των επαναλήψεων.

Η σύνταξη της δομής **for** είναι η παρακάτω:

```
for (αρχική τιμή αριθμητή; Συνθήκη επανάληψης; βήμα αύξησης ή μείωσης αριθμητή)
{
 κώδικας προς εκτέλεση;
}
```

Όπου αρχική τιμή αριθμητή είναι η αρχική τιμή του μετρητή, συνθήκη επανάληψης, μια λογική έκφραση που καθορίζει για ποιά τιμή του μετρητή θα σταματήσει η επανάληψη και βήμα η τιμή κατά την οποία αυξάνεται ή μειώνεται η τιμή του αριθμητή. Όσο η συνθήκη είναι αληθής(true), εκτελείται ο εμφωλευμένος κώδικας ανάμεσα στα άγκιστρα. Σε περίπτωση που ο εμφωλευμένος κώδικας είναι μια εντολή, τα άγκιστρα μπορούν να παραλειφθούν.

Παράδειγμα

Κώδικας	Αποτέλεσμα

<?php
for ($i=0; $i<10; $i++)
{
 echo "i: " . $i . "
";
}
?>
	i:0
	i:1
	i:2
	i:3
	i:4
	i:5
	i:6
	i:7
	i:8
	i:9

3.3.2 Η εντολή foreach

Η εντολή foreach χρησιμοποιείται κυρίως για την προσπέλαση ενός πίνακα. Σε κάθε επανάληψη προσπελαύνει ένα στοιχείο από τον πίνακα αρχίζοντας από το πρώτο και τελειώνοντας στο τελευταίο.

Έτσι αν $array είναι ο πίνακας και $value η τιμή του κάθε στοιχείου του πίνακα η σύνταξη της εντολής **foreach** είναι η παρακάτω:

```
foreach ($array as $value)
{
 κώδικας προς εκτέλεση;
}
```

.

Παράδειγμα

Κώδικας	Αποτέλεσμα

<?php
$a=array("Παρασκευή","Σάββατο","Κυριακή");
foreach ($a as $value)
{
echo $value . "
";
}
?>
	Παρασκευή
	
	Σάββατο
	
	Κυριακή

3.3.3 Η εντολή while

Η εντολή while εκτελεί το τμήμα εμφωλευμένου κώδικα όσο η συνθήκη, η οποία ελέγχεται πρώτα, είναι αληθής (true).

Η δομή της πρότασης **while** είναι η παρακάτω:

```
while (συνθήκη)
{
 κώδικας προς εκτέλεση;
}
```

Παράδειγμα

Κώδικας	Αποτέλεσμα

<?php
$i=0;
while($i<10)
{
 echo "i: " . $i . "
";
 $i++;
}
?>
	i:0
	i:1
	i:2
	i:3
	i:4
	i:5
	i:6
	i:7
	i:8
	i:9

3.3.4 Η εντολή do...while

Η εντολή do...while εκτελεί το τμήμα εμφωλευμένου κώδικα όσο η συνθήκη, η οποία ελέγχεται πρώτα, είναι αληθής (true).

Η σύνταξη της εντολής **do...while** είναι η παρακάτω:

```
do
{
 κώδικας προς εκτέλεση;
}
while (συνθήκη);
```

Παράδειγμα

Κώδικας	Αποτέλεσμα
<?php $i=0; do{ echo "i: " . $i . " "; $i++; } while($i<10) ?>	i:0 i:1 i:2 i:3 i:4 i:5 i:6 i:7 i:8 i:9

4 Δομές Δεδομένων

4.1 Πίνακες (arrays)

Για την προσωρινή αποθήκευση και διαχείριση πολλών δεδομένων στην κύρια μνήμη του υπολογιστή, η PHP όπως και οι άλλες γλώσσες υψηλού επιπέδου υποστηρίζουν σύνθετες δομές δεδομένων όπως είναι οι πίνακες (arrays). Οι πίνακες έχουν συνήθως τη δυνατότητα να αποθηκεύουν σε συνεχόμενες θέσεις μνήμης πολλά δεδομένα τα οποία και είναι πολύ εύκολα προσπελάσιμα και διαχειρίσιμα. Στην PHP, η συνάρτηση array() χρησιμοποιείται για την κατασκευή ενός πίνακα
array();

Σε ένα array μπορούμε να αποθηκεύσουμε ένα σύνολο από δεδομένα όπως ονόματα, αριθμούς, αντικείμενα κ.λπ. Αν για παράδειγμα έχουμε να αποθηκεύσουμε ένα σύνολο από μάρκες αυτοκινήτων και το όνομα του πίνακα είναι $car, τότε γράφουμε:

```
<?php
$car[0]="bmw";
$car[1]="honda";
$car[2]="mazda";
?>
```

Εναλλακτικά μπορούν να αποθηκευθούν δεδομένα σε έναν πίνακα με το εξής τρόπο.

```
<?php
$car = array("bmw","honda","mazda");
?>
```

Στην περίπτωση αυτή οι μεταβλητές $car[0], $car[1] και $car[] ονομάζονται στοιχεία του πίνακα. Το πλήθος των στοιχείων ενός πίνακα λέγεται μήκος του πίνακα. Για το παραπάνω παράδειγμα το μήκος είναι 3. Αν δεν γνωρίζετε το μήκος ενός πίνακα από την αρχή τότε μπορείτε να χρησιμοποιήσετε την εντολή count(). Στο παράδειγμα η count($car) θα επέστρεφε 3.

Για να εμφανιστεί η τιμή του πίνακα car στην $1^η$ θέση, χρησιμοποιείται για παράδειγμα η εντολή: echo $car[0];

Κώδικας	Αποτέλεσμα

<?php
for ($i=0; $i<3; $i++)
{
echo "Θέση " . ($i+1) .$car[$i]
	Θέση 1 bmw

" "; } ?>	Θέση 2 honda Θέση 3 mazda

4.1.1 Τύποι πινάκων

Στην PHP, υπάρχουν 3 τύποι πινάκων:

- **Πίνακες με απαρίθμηση** (**Indexed arrays)** – Πίνακες που χρησιμοποιούνται με έναν αριθμητικό δείκτη για τον καθορισμό της θέσης
- **Συσχετιζόμενοι Πίνακες (Associative arrays)** – Πίνακες που το κάθε στοιχείο είναι συσχετισμένο με ένα κλειδί.
- **Πολυδιάστατοι πίνακες (Multidimensional arrays)** – Πίνακες που περιέχουν πολλούς πίνακες

4.1.2 Προσπέλαση πίνακα με απαρίθμηση

Για να προσπελασθούν όλες τις τιμές ενός πίνακα με απαρίθμηση χρησιμοποιείται ένας βρόχος (loop) συνήθως με τη δομή for.

Παράδειγμα

Κώδικας	Αποτέλεσμα

<?php
$days[0]="Δευτέρα";
$days[1]="Τρίτη";
$days[2]="Τετάρτη";
$days[3]="Πέμπτη";
$days[4]="Παρασκευή";
 	
for($i=0; $i<5; $i++)
{
  echo($days[$i]." ");
}
?>
	Δευτέρα Τρίτη Τετάρτη Πέμπτη Παρασκευή

Στην περίπτωση που το μήκος του πίνακα δεν είναι γνωστό, χρησιμοποιείται η συνάρτηση count().

Παράδειγμα

```
<?php
for($i=0; $i<count($days); $i++)
{
  echo($days[$i]." ");
}
?>
```

Ένας άλλος πιο πρακτικός ίσως βρόχος είναι με τη χρήση του foreach().

Παράδειγμα

```
<?php

foreach($days as $val)
{
  echo($val." ");
}
?>
```

4.1.3 Συσχετιζόμενοι πίνακες (associative arrays)

Εκτός από τους πίνακες με απαρίθμηση που είδαμε παραπάνω, στην PHP χρησιμοποιούνται οι συσχετιζόμενοι πίνακες όπου το κάθε στοιχείο είναι συσχετισμένο με ένα κλειδί.

'Ενας συσχετιζόμενος πίνακας έχει την παρακάτω μορφή.

```
<?php
$salary=array();
$salary["nikos"]=525;
$salary["pavlos"]=732;
$salary["anna"]=920;
$salary["yannis"]=1024;
$salary["efi"]=735;
?>
```

Στο παραπάνω παράδειγμα κάθε όνομα έχει συσχετισθεί με έναν μισθό. Έτσι για να εμφανιστεί ο μισθός του Γιάννη αρκεί να εκτελεστεί η εντολή: echo($salary["yannis"]);

Η πρώτη τιμή π.χ. " yannis " λέγεται κλειδί (key) ενώ η συσχετισμένη τιμή, που στο παράδειγμα είναι το , λέγεται τιμή (value)

Για τον ίδιο συσχετισμένο πίνακα η σύντομη γραφή είναι:

```
<?php
$salary=array("nikos"=>525, "pavlos"=>732, "anna"=>920,
"yannis"=>1024, "efi"=>735 );
?>
```

Για να γίνει προσπέλαση των στοιχείων σε ένα συσχετισμένο πίνακα και να χρησιμοποιηθούν και το κλειδί αλλά και η τιμή χρησιμοποιείται ο βρόχος foreach().

Παράδειγμα

Κώδικας	Αποτέλεσμα

<?php
$salary=array("nikos"=>525, "pavlos"=>732,
"anna"=>920, "yannis"=>1024, "efi"=>735);
 	
foreach($salary as $key => $val)
{
  echo "$key – $val
";
}
?>
	nikos - 525
	pavlos - 732
	anna -9 20
	yannis - 1024
	efi – 735

4.2 Πολυδιάστατοι πίνακες στην PHP

Ένας πολυδιάστατος πίνακας είναι στην πραγματικότητα ένας πίνακας που σαν στοιχεία περιέχει έναν ή περισσότερους πίνακες. Η PHP μπορεί και διαχειρίζεται πολυδιάστατους πίνακες που περιέχουν 2, 3 ή και περισσότερα επίπεδα πινάκων. Πέραν όμως των τρισδιάστατων πινάκων, οι πίνακες γίνονται δύσκολα διαχειρίσιμοι για τους προγραμματιστές.

Το πλήθος των διαστάσεων καθορίζει και το πλήθος των διαφορετικών δεικτών απαρίθμησης που απαιτούνται για την προσπέλαση των στοιχείων. Συνεπώς για

πίνακες δύο διαστάσεων απαιτούνται δύο μεταβλητές – δείκτες απαρίθμησης ενώ αντίστοιχα για τρεις διαστάσεις απαιτούνται 3 μητεβλητές – απαριθμητές.

4.2.1 Δισδιάστατοι Πίνακες

Ένας δισδιάστατος πίνακας είναι ένας πίνακας πινάκων. Για παράδειγμα θα μπορούσε να ήταν ένας πίνακας σαν το παράδειγμα:

Μάρκα Τιμή Απόθεμα

Honda 22000 18

BMW 25000 13

Mazda 20000 2

Τα παραπάνω δεδομένα θα μπορούσαν να αποθηκευθούν σε ένα πίνακα στην PHP ως εξής:

```
$cars = array
  (
  array("Honda ",22000,18),
  array("BMW",25000,13),
  array("Mazda",20000,2)  );
```

Ο πίνακας cars είναι δισδιάστατος γιατί κάθε στοιχείο του πίνακα περιέχει ένα array και συνεπώς χρειάζονται δύο δείκτες απαρίθμησης, ένας για τη γραμμή και ένας για τη στήλη.

Για την προσπέλαση του πίνακα θα πρέπει να γίνεται προσδιορισμός της γραμμής και στήλης του πίνακα:

Κώδικας	Αποτέλεσμα
`<?php` `echo $cars[0][0].": με τιμή:` `".$cars[0][1].", σε απόθεμα:` `".$cars[0][2].". ";` `echo $cars[1][0].": με τιμή:` `".$cars[1][1].", σε απόθεμα:` `".$cars[1][2].". ";` `echo $cars[2][0].": με τιμή:` `".$cars[2][1].", σε απόθεμα:` `".$cars[2][2].". ";?>`	Honda: με τιμή: 22000, σε απόθεμα: 18 BMW: με τιμή: 25000, σε απόθεμα: 13 Mazda: με τιμή: 20000, σε απόθεμα: 2

5 Συναρτήσεις (Functions)

Μια από τις τεχνικές προγραμματισμού που υποστηρίζεται από την PHP είναι ο τμηματικός προγραμματισμός ο οποίος και υλοποιείται όπως και στην περίπτωση της C και Java με τη χρήση συναρτήσεων. Οι συναρτήσεις αποτελούν υποπρογράμματα, τμήματα δηλαδή προγράμματος τα οποία καλούνται για να επιτελέσουν μια ανεξάρτητη και αυτόνομη λειτουργία του προγράμματος. Κάθε συνάρτηση δεν εκτελείται παρα μόνο όταν καλείται από κάποιο άλλο τμήμα κώδικα. Η γενική μορφή μιας συνάρτησης είναι:

```
function FunctionName()
{
 ...
}
```

Το όνομα μιας συνάρτησης πρέπει να είναι μοναδικό όνομα (π.χ.: **myFunc**) και ακολουθείται από ένα ζεύγος παρενθέσεων (π.χ.: **function myFunc()**) όπου αναφέρονται οι παράμετροι που δέχεται η συνάρτηση ως είσοδο, χωρίς η ύπαρξη παραμέτρων να είναι πάντα υποχρεωτική.

Το τμήμα κώδικα της συνάρτησης περιέχεται σε άγκιστρα { }. (π.χ.: **function myFunc() { . . . }**). Η συνάρτηση καλείται με το όνομά της (π.χ.: **myFunc()**) όσες φορές είναι απαραίτητο από οποιοδήποτε τμήμα προγράμματος ενώ ένα έγγραφο php μπορεί να περιέχει ή να καλεί περισσότερες από μία συναρτήσεις

5.1 Παράμετροι

Μια συνάρτηση μπορεί να εκτελεί κάποια εργασία χωρίς να είναι δέχεται κάποια είσοδο σαν τιμή με την μορφή παραμέτρου από κάποιο άλλο τμήμα προγράμματος. Παρόλα αυτά η χρήση των παρενθέσεων κατά την κλήση είναι απαραίτητη ακόμα κι αν αυτές είναι κενές.

Στο ακόλουθο απλό παράδειγμα έχει οριστεί μια συνάρτηση με το όνομα HelloWorld **()** η οποία εκτελείται κάθε φορά που καλείται.

Παράδειγμα

Κώδικας	Αποτέλεσμα
<?php function HelloWorld()	Hello World

{
echo("Hello World ");
}
HelloWorld ();
?>	

Ο τμηματικός προγραμματισμός έχει ως βασικό στοιχείο την αυτονομία των τμημάτων κώδικα. Γι αυτό και όπως αναφέρθηκε στην παράγραφο για την εμβέλεια των μεταβλητών, η κάθε μεταβλητή είναι τοπική (εκτός κι αν δηλωθεί ως global μέσα στη συνάρτηση) και η εμβέλειά της περιορίζεται στο τμήμα κώδικα στο οποίο έχει δηλωθεί. Για να μπορέσει να περάσει μια τιμή μιας μεταβλητής στο τμήμα κώδικα που είναι μέσα στη συνάρτηση, χρησιμοποιούνται παράμετροι, δηλαδή μεταβλητές οι οποίες επιτρέπουν το πέρασμα τιμών από το ένα τμήμα προγράμματος στο άλλο. Οι παράμετροι καταχωρούνται μέσα στις παρενθέσεις, δίπλα από το όνομα της συνάρτησης και δηλώνονται επίσης μέσα στις παρενθέσεις κατά τη δήλωση της συνάρτησης. Μπορούν να δηλωθούν περισσότερες από μία παραμέτροι οι οποίες χωρίζονται με κόμμα. Το όνομα κάθε παραμέτρου ακολουθεί τους κανόνες ονοματολογίας που ισχύει και για τις κοινές μεταβλητές.

Κώδικας	Αποτέλεσμα

<?php
function Hello($name)
{
echo("Hello ".$name);
}
Hello(“George”);
?>
	Hello George

Στη συνάρτηση του παραδείγματος η τιμή της παραμέτρου είναι τύπου string αλλά μπορεί να είναι και όποιου άλλου τύπου που υποστηρίζει η php.

Στο παρακάτω παράδειγμα η συνάρτηση **square** υπολογίζει το τετράγωνο ενός αριθμού. Έτσι, αν κληθεί ως **square (10)** θα δώσει σαν αποτέλεσμα το 100.

Κώδικας	Αποτέλεσμα

<?php
function square($x)
{
echo($x*$x);
}
square(10)
	100

?>	

Επειδή η τιμή της παραμέτρου είναι αριθμός, δεν μπαίνει σε εισαγωγικά

5.2 Συναρτήσεις που επιστρέφουν τιμές

Μια συνάρτηση μπορεί να επιστρέφει μια τιμή ως αποτέλεσμα κάποιας επεξεργασίας. Στην περίπτωση αυτή χρησιμοποιείται η εντολή **return**.

Στο παρακάτω παράδειγμα η συνάρτηση **getAbsolute()** επιστρέφει την απόλυτη τιμή ενός αριθμού που περνάει ως παράμετρος.

Κώδικας	Αποτέλεσμα

<?php
function getAbsolute($x)
{
If($x<0)
$z=-1*$x;
Else
$z=$x;
return $z;
}
 	
$x=-2;
echo getAbsolute($x);
?>
	2

Η εντολή return μπορεί επίσης να χρησιμοποιηθεί για να διακοπεί η εκτέλεση της συνάρτησης.

5.3 Προεπιλεγμένη τιμή παραμέτρου

Η PHP δίνει τη δυνατότητα να οριστεί κατά τη δήλωση της συνάρτησης μια προεπιλεγμένη τιμή για τις παραμέτρους. Αν η συνάρτηση κληθεί χωρίς κάποια τιμή για την παράμετρο τότε θα χρησιμοποιηθεί η προεπιλεγμένη.

Κώδικας	Αποτέλεσμα

<?php
function setValue($x = 0) {
    echo "Η τιμή είναι : $x
";
}
 	
setValue (350);
setValue (); // default τιμή 0
?>
	Η τιμή είναι : 350
	
	Η τιμή είναι : 0

5.4 Συμβολοσειρές (Strings)

Μια συμβολοσειρά (string) είναι μια ακολουθία χαρακτήρων, όπως για παράδειγμα η ακολουθία "Hello world!". Η PHP παρέχει μια σειρά από τελεστές και έτοιμες συναρτήσεις για την διαχείριση των συμβολοσειρών. Βασικός τελεστλης που χρησιμοποιείται πολύ συχνα είναι ο τελεστής «.» ο οποίος χρησιμοποιείται για την συνένωση συμβολοσειρών μεταξύ τους.

Κώδικας	Αποτέλεσμα
<?php $x=”test”; $y=”my “; echo $y.$x; ?>	My test

5.5 Συναρτήσεις Συμβολοσειρών

- **char()**

η char() επιστρέφει τον αντίστοιχο ascii χαρακτήρα ενός ακέραιου αριθμού. Η σύνταξη της συνάρτησης είναι: *string* **chr** (*int*)

Κώδικας	Αποτέλεσμα
<?php echo chr(97); ?>	a

- **ord()**

Η ord(), αντίθετα από την char(), επιστρέφει τον ακέραιο αριθμό που αντιστοιχεί στον ascii χαρακτήρα και συντάσσεται ως εξής: *int* **ord** (*string*)

Κώδικας	Αποτέλεσμα
<?php echo ord('a'); ?>	97

- **explode()**

Η συνάρτηση αυτή δημιουργεί ένα array που περιέχει τα στοιχεία μιας συμβολοσειράς τα οποία είναι χωρισμένα με έναν ή περισσότερους διαχωριστικούς χαρακτήρες. Η σύνταξη της συνάρτησης είναι: *array* **explode** (*string*,*string*)

Κώδικας	Αποτέλεσμα
<?php $names = "john,chris,betty"; $pieces = explode(",", $names); echo $pieces[1]; ?>	chris

- **implode()**

Δημιουργεί ένα string που περιέχει τα στοιχεία ενός πίνακα (array) με ανάμεσά τους έναν διαχωριστικό χαρακτήρα. Η σύνταξη της συνάρτησης είναι: *string* **implode** (*string*, *array*)

Κώδικας	Αποτέλεσμα
<?php $arr = array('john', ' chris ', ' betty '); $comma_separated = implode(",", $arr); echo $comma_separated; ?>	john,chris,betty

- **ltrim()**

Η συνάρτηση επιστρέφει ένα string αφαιρώντας από ένα αρχικό string από τα αριστερά κενούς χαρακτήρες ή άλλους χαρακτήρες. Η σύνταξη της συνάρτησης είναι: *string* **ltrim** (*string*[,*string*])

Κώδικας	Αποτέλεσμα
<?php $hello = "***Hello World"; echo ltrim($hello, "*"); ?>	Hello World

- **rtrim()**

αντίστοιχα η ltrim(), επιστρέφει ένα string από ένα αρχικό του οποίου έχει αφαιρέσει από τα αριστερά κενούς χαρακτήρες ή άλλους χαρακτήρες.
Η σύνταξη της συνάρτησης είναι: *string* **rtrim** (*string*[,*string*])

- **trim()**

Συνδυάζοντας ταυτόχρονα την ltrim() και την rtrim(), η συνάρτηση επιστρέφει ένα string από ένα αρχικό του οποίου έχει αφαιρέσει από τα δεξιά και τα αριστερά τους κενούς ή ‘άλλους χαρακτήρες . Η σύνταξη της συνάρτησης είναι: *string* **trim** (*string*[,*string*])

Κώδικας	Αποτέλεσμα
<?php $hello = "***Hello World*****"; echo trim($hello, "*"); ?>	Hello World

- **str_getcsv()**

Παρόμοια με την explode() η συνάρτηση επιστρέφει ένα array που περιέχει τα στοιχεία μιας συμβολοσειράς csv (comma separated values) όπου τα στοιχεία είναι διαχωρισμένα συγκεκριμένα με κόμα. Η σύνταξη της συνάρτησης είναι: *array* **str_getcsv** (*string*)

Κώδικας	Αποτέλεσμα
<?php $csv = " john,chris,betty "; $names=str_getcsv($csv); echo $names[0]; ?>	john

- **str_replace()**

Επιστρέφει μια συμβολοσειρά αντικαθιστώντας ένα τμήμα συμβολοσειράς με μια άλλη μέσα σε ένα κείμενο. Η σύνταξη της συνάρτησης είναι: *string* **str_replace**(*string*, *string*, *string*)

Κώδικας	Αποτέλεσμα
<?php $text = str_replace("friend", "best friend", "Nick is my friend"); echo $text; ?>	Nick is my best friend

Επίσης, αντί αντικατάστασης ενός μόνο string μπορεί να γίνει αντικατάσταση στοιχείων array με την ακόλουθη σύνταξη της συνάρτησης: string str_replace(array, array, string)

Κώδικας	Αποτέλεσμα
<?php $phrase = "You should eat fruits, vegetables, and fiber every day."; $healthy = array("fruits", "vegetables", "fiber"); $yummy = array("pizza", "beer",	You should eat pizza, beer, and ice cream every day.

"ice cream"); $newphrase = str_replace($healthy, $yummy, $phrase); echo $newphrase; ?>	

- **str_shuffle()**

Η συνάρτηση παράγει μια συμβολοσειρά αφού ανακατέψει τυχαία τα στοιχεία μιας συμβολοσειράς. Η σύνταξη της συνάρτησης είναι: *string* **str_shuffle**(*string*)

Κώδικας	Αποτέλεσμα
<?php $str = 'hello'; $shu = str_shuffle($str); echo $shu; ?>	loleh

- **str_split()**

Η συνάρτηση διαχωρίζει μια συμβολοσειρά σε χαρακτήρες και επιστρέφει το αποτέλεσμα σε ένα array όπου το κάθε στοιχείο του πίνακα είναι και ένας χαρακτήρας. Η σύνταξη της συνάρτησης είναι: *array* **str_split**(*string*)

Κώδικας	Αποτέλεσμα
<?php $str = 'world'; $arr1 = str_split($str); print_r($arr1); ?>	Array ([0] => w [1] => o [2] => r [3] => l [4] => d)

Επίσης, μπορεί να διαχωρίσει τη συμβολοσειρά ανά κάποιο πληθος χαρακτήρων και να τα επιστρέψει σε ένα array με την ακόλουθη σύνταξη της συνάρτησης: *array* **str_split**(*string*, *int*)

```
<?php
 $str = "tictactoe";
 $arr = str_split($str, 3);
 print_r($arr);
?>
```

Array ([0] => tic [1] => tac [2] => toe [3])

- **strcmp()**

Η σύνταξη της συνάρτησης είναι: *int* **strcmp**(*string*, *string*) και χρησιμοποιείται για να συγκρίνει δύο συμβολοσειρές και επιστρέφει

0 αν είναι ίδιες

αρνητικό αριθμό αν η πρώτη είναι μικρότερη της δεύτερης

θετικό αριθμό αν η πρώτη είναι μεγαλύτερη της δεύτερης.

Κώδικας	Αποτέλεσμα
<?php $str1="chelo"; $str2="hello"; echo strcmp($str1, $str2); ?>	1

- **strlen()**

Η σύνταξη της συνάρτησης είναι: *int* **strlen**(*string*) και επιστρέφει το μήκος μιας συμβολοσειράς.

Κώδικας	Αποτέλεσμα
<?php $str = 'abcdef'; echo strlen($str); ?>	6

- **strrev()**

Επιστρέφει μια συμβολοσειρά με την αντίστροφη σειρά της αρχικής συμβολοσειράς. Η σύνταξη της συνάρτησης είναι: *string* **strrev**(*string*)

Κώδικας	Αποτέλεσμα
<?php echo strrev("Hello world!"); ?>	!dlrow olleH

- **strtolower()**

Η σύνταξη της συνάρτησης είναι: *string* **strtolower**(*string*) και μετασχηματίζει τα στοιχεία μιας συμβολοσειράς από κεφαλαία σε πεζά γράμματα.

Κώδικας	Αποτέλεσμα
<?php $str = "Hello world"; $str = strtolower($str); echo $str; ?>	hello world

- **strtoupper()**

Η σύνταξη της συνάρτησης είναι: *string* **strtoupper**(*string*) και μετασχηματίζει τα στοιχεία μιας συμβολοσειράς από πεζά σε κεφαλαία γράμματα.

Κώδικας	Αποτέλεσμα
<?php $str = "Hello world"; $str = strtoupper ($str);	HELLO WORLD

echo $str; ?>	

- **substr()**

Επιστρέφει μέρος μιας συμβολοσειράς ξεκινώντας από μια θέση στη συμβολοσειρά και προχωρώντας κάποιο πλήθος θέσεων. Η σύνταξη της συνάρτησης είναι: *string* **substr**(*string*, *int*, *int*)

```
<?php
 echo substr('abcdef', 1, 3);
?>
```

6 Αρχεία

6.1 Άνοιγμα αρχείου με fopen()

Η καταλληλότερη μέθοδος για το άνοιγμα αρχείων προς διαχείριση είναι η συνα΄ρτηση fopen(). Υποθέτοντας ότι το αρχείο κειμένου που πρόκειται να χρησιμοποιήσουμε για άνοιγμα είναι το "mytext.txt", η σύνταξη της fopen είναι η εξής:

Η πρώτη παράμετρος της fopen() περιέχει το όνομα του αρχείου που πρόκειται να ανοιχθεί ενώ η δεύτερη παράμετρος προσδιορίζει την κατάσταση στην οποία θα ανοιχθεί το αρχείο. Στο ακόλουθο παράδειγμα αν το αρχείο δεν μπορέσει να ανοιχθεί σε κατάσταση ανάγνωσης, τότε θα παραχθεί ένα μήνυμα σφάλματος.

```
<?php
$myfile = fopen("mytext.txt", "r") or die("Unable to open file!");
echo fread($myfile,filesize("mytext.txt"));
fclose($myfile);
?>
```

Ένα αρχείο μπορεί να ανοιχτεί στις ακόλουθες καταστάσεις:

Κατάσταση	**Περιγραφή**
r	**Ανοίγει ένα αρχείο μόνο για ανάγνωση**. Ο δείκτης τους αρχείου ξεκινά από την αρχή του αρχείου
w	**Ανοίγει ένα αρχείο μόνο για εγγραφή**. Διαγρ'αφει τα περιεχόμενα του αρχείου αν αυτό υπάρχει διαφορετικά δημιουργεί ένα νέο. Ο δείκτης τους αρχείου ξεκινά από την αρχή του αρχείου
a	**Ανοίγει ένα αρχείο για προσθήκη**. Το υπάρχον αρχείο παραμένει ίδιο και ο δείκτης του αρχείου για εγγραφή τοποθετείται στο τέλος του κειμένου. Αν το αρχείο δεν υπάρχει δημιουργείται ένα νέο.
x	**Δημιουργεί ένα νέο αρχείο για εγγραφή**. Επιστρέφει την τιμή FALSE και ένα σφάλμα αν το αρχείο υπάρχει διαφορετικά το αρχείο δημιουργείται.
r+	**Ανοίγει ένα αρχείο για ανάγνωση και εγγραφή**. Ο δείκτης του αρχείου τοποθετείται στην αρχή
w+	**Ανοίγει ένα αρχείο για ανάγνωση και εγγραφή**. Διαγράφει τα περιεχόμενα του αρχείου η δημιουργεί ένα νέο αν αυτό δεν υπάρχει. Ο δείκτης τοποθετείται στην αρχή του αρχείου.
a+	**Ανοίγει ένα αρχείο για ανάγνωση και εγγραφή**. Το υπάρχον αρχείο

	παραμένει ίδιο και ο δείκτης του αρχείου για εγγραφή τοποθετείται στο τέλος του κειμένου. Αν το αρχείο δεν υπάρχει δημιουργείται ένα νέο.
x+	**Δημιουργεί ένα νέο αρχείο για ανάγνωση και εγγραφή**. Επιστρέφει την τιμή FALSE και ένα σφάλμα αν το αρχείο υπάρχει διαφορετικά το αρχείο δημιουργείται.

6.2 Ανάγνωση αρχείου με την fread()

Η συνάρτηση fread() διαβάζει ένα ανοιχτό αρχείο. Η πρώτη παράμετρος περιεχει το όνομα του αρχείου που πρόκειται να διαβασθεί και η δεύτερη παράμετρος προσδιορίζει το μέγιστο πλήθος bytes που πρόκειται να διαβασθούν. Η σκόλουθη εντολή διαβάζει όλο το κείμενο του αρχείου mytext.txt.

```
fread($myfile,filesize("mytext.txt"));
```

6.3 Κλείσιμο αρχείου με την fclose()

Η συνάρτηση fclose() χρησιμοποιείται για το κλείσιμο ενός ανοικτού αρχείου. Είναι ορθή τακτική προγραμματισμού να κλείνονται τα ανοικτά αρχεία αμέσως μετά την επεξεργασία τους για να αποφεύγεται η άσκοπη σπατάλη πόρων του υπολογιστή. Η σύνταξη της fclose() περιλαμβάνει το όνομα του αρχείου ή της μεταβλητης που περιέχει το όνομα που πρόκειται να κλείσει:

```
<?php
$myfile = fopen("mytext.txt", "r");
fclose($myfile);
?>
```

6.4 Ανάγνωση μιας μόνο γραμμής με την fgets()

Η συνάρτηση fgets() χρησιμοποιείται για να διαβάσει ένα αρχείο κατά γραμμή. Μετά από κάθε εκτέλεση της συναρτησης, ο δείτκης του αρχείου μεταφέρεται στο τέλος της γραμμής. Στο παρακάτω παράδειγμα το αρχείο ανοίγει και η συνάρτηση διαβάζει την πρώτη γραμμή:

```
<?php
$myfile = fopen("mytext.txt", "r") or die("Unable to open file!");
echo fgets($myfile);
fclose($myfile);
?>
```

6.5 Έλεγχος τέλους αρχείου με την End-Of-File - feof()

Η συνάρτηση feof() ελέγχει αν το αρχείο έχει φτάσει στο τέλος του ("end-of-file" - EOF). Χρησιμοποιείται κυρίως στην περίπτωση που πρόκειται να διαβασθούν δεδομένα ενός αρχείου άγνωστου μεγέθους.

```
<?php
$myfile = fopen("mytext.txt", "r") or die("Unable to open file!");
// Output one line until end-of-file
while(!feof($myfile)) {
  echo fgets($myfile) . "<br>";
}
fclose($myfile);
?>
```

6.6 Ανάγνωση μεμονωμένου χαρακτήρα - fgetc()

Η συνάρτηση fgetc() χρησιμοποιείται για να διαβάσει έναν μεμονωμένο χαρακτήρα από ένα αρχείο.

Στο παρακάτω παράδειγμα διαβάζεται το αρχείο "mytext.txt" χαρακτήρα – χαρακτήρα μέχρι το τέλος του αρχείου.

παράδειγμα

```
<?php
$myfile = fopen("mytext.txt", "r") or die("Unable to open file!");
while(!feof($myfile)) {
  echo fgetc($myfile);
}
fclose($myfile);
?>
```

7 Αντικειμενοστραφής προγραμματισμός

Η PHP υποστηρίζει εκτός από τον τμηματικό και συναρτησιακό προγραμματισμό, προγράμματα γραμμένα σε αντικειμενοστραφή προγραμματισμό. Το βασικό στοιχείο αυτού είναι η κλάση, μια δομή η οποία αναπαριστά και περιγράφει κάποια έννοια ή οντότητα του προβλήματος.

Η κλάση περιγράφει τις προδιαγραφές με τις οποίες θα δημιουργηθούν αντικείμενα του προβλήματος κατά την εκτέλεση ενός προγράμματος.

7.1 Κλάσεις

Οι κλάσεις δηλώνονται με τη χρήση της παρακάτω σύνταξης:

class *ονομα_κλάσης* { }
Για παράδειγμα: class car { ... }

Μια κλάση περιέχει έναν συνδυασμό από μεταβλητές (properties), συναρτήσεις (methods) καθώς και μια μέθοδο εγκατάστασης ή κατασκευαστή (constructor) και καταστροφής (destructor). Οι μεταβλητές και οι συναρτήσεις της κλάσης λέγονται μέλη (members) της κλάσης.

7.1.1 Η μεταβλητή –μέλος και συνάρτηση - μέλος

Μια μεταβλητή – μέλος δηλώνεται με τον τροποποιητή (public, private, protected) ακολουθούμενο από το όνομα της μεταβλητής

Για παράδειγμα: public $speed;

Μια συνάρτηση – μέλος δηλώνεται ξεκινώντας από τον τροποποιητή (public, private, protected), τον προσδιοριστή function και το όνομα της συνάρτησης ακολουθούμενο από λίστα παραμέτρων μέσα σε παρενθέσεις και το τμήμα κώδικα που πρέπει να εκτελείται μέσα σε {}.

```
function get_age($birth_year)
{
  $current_year=date('Y');
  return $current_year – $birth_year;
}
```

Αν δεν δηλωθεί ο τροποποιητής τότε ο προκαθορισμένος είναι: public

7.1.2 Ο κατασκευαστής

Ο κατασκευαστής είναι μια συνάρτηση και έχει το όνομα __construct() είναι προεπιλεγμένα public. Αντίθετα με τις άλλες συναρτήσεις δεν επιστρέφει καμία τιμή και κάθε φορά που εκτελείται αυτόματα δημιουργείται ένα νέο αντικείμενο

```
<?php
class Car
{
  private $age;

  function __construct()
  {
    $age=0;
  }

  function get_age()
  {
    return $age;
  }
}
?>
```

Κάθε κλάση μπορεί επίσης να περιέχει έναν καταστροφέα ο οποίος καλείται κάθε φορά που ένα αντικείμενο τελειώνει τον σκοπό του. Ο καταστροφέας είναι ορισμένος προεπιλεγμένα, χωρίς να χρειάζεται να αναφερθεί αλλά οι προγραμματιστές έχουν τη δυνατότητα να προσθέσουν τον δικό τους προσαρμοσμένο καταστροφέα. Ο καταστροφέας είναι, όπως και ο κατασκευαστής, μια public συνάρτηση με την παρακάτω σύνταξη.

```
function __destruct()
{

}
```

7.2 Αντικείμενα (Objects)

Χρησιμοποιώντας τις προδιαγραφές που ορίζονται από τις κλάσεις μπορούν να δημιουργηθούν αντικείμενα τα οποία είναι συγκεκριμένα στιγμιότυπα μιας κλάσης. Κάθε αντικείμενο δημιουργείται με την εντολή new και την κλήση της συνάρτησης κατασκευής.

Όταν γίνεται αναφορά στις μεταβλητές της κλάσης από μια συνα΄ρτηση, τότε χρησιμοποιούμε ο τελεστής $this ακολουθούμενος από τον τελεστή ->. Δηλαδή αν η

μεταβλητή – μέλος είναι η $speed τότε αυτή αναφέρεται ως $this->speed. Για παράδειγμα $this->speed=200 και όχι $speed=200.

Παράδειγμα

Στο παρακάτω παράδειγμα όταν δημιουργείται το αντικείμενο $Stathis καλείται αυτόματα ο κατασκευαστής ο οποίος θέτει στη μεταβλητή $current_year την τιμή για το τρέχον έτος. Μετά τη δημιουργία του αντικειμένου, το αντικείμενο καλεί τη συνάρτηση get_age με την κατάλληλη παράμετρο. Προσέξτε ότι χρησιμοποιείται το τελεστής -> όταν καλείται μια συνάρτηση ή μεταβλητή.

```
<?php
class Car
{
  private $age;
  private $speed;


  function __construct()
  {
    $this->age=0;
    $this->speed=200;

  }

  function get_age($age)
  {
    return $$this->age;
  }
}

$Honda1 = new Car();
$age = $Honda1->get_age();
echo $age;
?>
```

Αποτέλεσμα εκτέλεσης

0

Παράδειγμα

Στο παρακάτω παράδειγμα χρησιμοποιείται ο κατασκευαστής για να υπολογίσει την ηλικία και να την αποθηκεύσει στη μεταβλητή $age

```
<?php
class Person
{
  private $age=0;

  function __construct($year)
  {
    $this->age=date('Y') - $year;
  }

  function get_age()
  {
    return $this->age;
  }
}

$Stathis = new Person(1987);
$Anna= new Person(1991);

$Stathis_age=$Stathis->get_age();
$Anna_age=$Anna->get_age();

echo "Ο Στάθης είναι $Stathis_age ετών";
echo "<br />";
echo "Η Άννα είναι $Anna_age ετών";
?>
```

Αποτέλεσμα εκτέλεσης

Ο Στάθης είναι 25 ετών
Η Άννα είναι 21 ετών

7.3 Κληρονομικότητα (Inheritance)

7.3.1 Γενικές αρχές για κληρονομικότητα και υποκλάσεις

- Μια κλάση που παράγεται από μια άλλη κλάση καλείται υποκλάση (subclass) ή παραγόμενη κλάση (derived) ή θυγατρική κλάση (child).

- Η κλάση από την οποία παράγεται μια κλάση λέγεται υπερκλάση (superclass) ή γονική κλάση (parent) ή βασική κλάση (base).
- Μια υποκλάση έχει τις δικές της μεταβλητές, συναρτήσεις και κατασκευαστή όπως κάθε άλλη κλάση.
- Μια υποκλάση κληρονομεί (inherit) τις public και protected μεταβλητές και συναρτήσεις της υπερκλάσης αλλά όχι τον κατασκευαστή, μπορεί όμως να έχει πρόσβαση σε αυτόν.
- Μια υποκλάση δεν έχει άμεση πρόσβαση στα private μέλη της υπερκλάσης αλλά έμμεσα μέσω των public μεθόδων της υπερκλάσης.
- Για να δηλώσουμε ότι μια κλάση, έστω Manager είναι υποκλάση ή κληρονομεί την κλάση έστω Employee, γράφουμε: class Manager extends Employee
- Μια κλάση μπορεί να κληρονομεί μόνο μία άλλη κλάση (δεν ισχύει η πολυκληρονομικότητα).
- Μια κλάση μπορεί να κληρονομεί μια κλάση η οποία μπορεί να κληρονομεί μια άλλη κλάση ... κ.λπ.
- Όταν καλείται μια συνάρτηση ή ο κατασκευαστής μιας υπερκλάσης τότε μπαίνει και το πρόθεμα parent::

Παράδειγμα

```
<?php
class Person
{
  private $firstName;
  private $lastName;

  function __construct($fn, $ln)
  {
    $this->firstName=$fn;
    $this->lastName=$ln;
  }

  function getFirstName()
  {
    return $this->firstName;
  }

  function getLastName()
  {
    return $this->lastName;
  }

  function setFirstName($fn)
```

```
  {
    $this->firstName=$fn;
  }

  function setLastName($ln)
  {
    $this->lastName=$ln;
  }
}

class Student extends Person
{
  private $am;

  function __construct($fn, $ln, $a)
  {
    parent::setFirstName($fn);
    parent::setLastName($ln);
    $this->am=$a;
  }

  function setAm($a)
  {
    $this->am=$a;
  }

  function getAm()
  {
    return $this->am;
  }
}

$Roula = new Student("Ρούλα", "Παπαρούλα", 3456);

echo $Roula->getFirstName();
echo "<br />";
echo $Roula->getLastName();
echo "<br />";
echo $Roula->getAm();
?>
```

Αποτέλεσμα εκτέλεσης

Ρούλα
Παπαρούλα
3456

7.4 Υπερκάλυψη (Overriding)

Γνωρίζουμε ότι όταν μια κλάση κληρονομεί μια άλλη κλάση τότε η υποκλάση που κληρονομεί την υπερκλάση μπορεί να χρησιμοποιήσει τις κληρονομημένες public και protected συναρτήσεις.

Υπάρχουν όμως περιπτώσεις όπου η υποκλάση χρειάζεται να χρησιμοποιήσει μια συνάρτηση με το ίδιο όνομα και παραμέτρους της υπερκλάσης (δηλαδή με την ίδια υπογραφή συνάρτησης). Σε αυτή την περίπτωση η συνάρτηση δηλώνεται και στην υποκλάση με το δικό της σώμα (κώδικα).

Έτσι, όταν χρειάζεται να καλέσουμε τη συνάρτηση έστω: get_name() της υπερκλάσης τότε γράφουμε: parent::get_name(); ενώ για τη get_name() της υποκλάσης γράφουμε: $this->get_name();.

Η υπερκάλυψη ισχύει και για τις μεταβλητές.

Παράδειγμα

```
<?php

class Person
{
  private $name;

  function get_name()
  {
    return $this->name;
  }

  function set_name($fn)
  {
    $this->name=$fn;
  }
}

class Student extends Person
{
  private $name;
```

```
  function __construct($fn)
  {
    $this->set_name($fn);
  }

  function get_name()
  {
    return $this->name;
  }

  function set_name($fn)
  {
    $this->name=$fn;
  }
}

$Roula = new Student("Ρούλα");
echo $Roula->get_name();

?>
```

Αποτέλεσμα εκτέλεσης

Ρούλα

7.5 Μεταβλητές και συναρτήσεις static

Ένα μέλος (μεταβλητή ή συνάρτηση) μιας κλάσης μπορεί να χαρακτηριστεί ως static.

Κάθε static μέλος έχει δύο βασικά χαρακτηριστικά.

- Η πρόσβασή του είναι άμεση χωρίς να χρειαστεί πρώτα να δημιουργηθεί κάποιο αντικείμενο.
- Όλα τα αντικείμενα που έχουν δημιουργηθεί με βάση μια συγκεκριμένη κλάση, μοιράζονται τα static μέλη της κλάσης. Δηλαδή μια static μεταβλητή για παράδειγμα είναι κοινή για όλα τα αντικείμενα της κλάσης.

Έτσι, αν για παράδειγμα αλλάξει η τιμή της μεταβλητής, τότε αυτή η τιμή περνάει σε όλα τα αντικείμενα που έχουν δημιουργηθεί με βάση αυτή την κλάση.

Σε ένα πιο πρακτικό παράδειγμα μπορεί να έχουμε μια κλάση έστω Product που περιέχει μια static μεταβλητή έστω $fpa στην οποία αποθηκεύεται η τιμή του Φ.Π.Α. και η οποία είναι ίδια για όλα τα προϊόντα. Αν τώρα αλλάξει η τιμή του φπα, απλά

αλλάζουμε την τιμή της static $fpa και επειδή είναι κοινή για όλα τα προϊόντα, όλα τα προϊόντα έχουν αυτόματα ενημερωθεί.

Ένα static μέλος μπορεί να είναι public, private ή ptotected.

Τέλος, η αναφορά σε ένα static μέλος, μέσα από την κλάση, γίνεται με τη λέξη self:: αντί του $this->.

Σε νεότερες εκδόσεις της php μπορείτε να χρησιμοποιήσετε και την static:: αντί της self::.

Παράδειγμα

Το παρακάτω παράδειγμα δείχνει πως μπορείτε να έχετε πρόσβαση σε μια static μεταβλητή χρησιμοποιώντας το όνομα της κλάσης και όχι με τη χρήση αντικειμένου.

```
<?php
class Product
{
  static $fpa=23;
}
echo Product::$fpa;
?>
```

23

Χρησιμοποιούμε τον τελεστή :: για την πρόσβαση της μεταβλητής.

Επειδή χρησιμοποιούμε το όνομα της κλάσης και μετά το όνομα του static μέλους, τα static μέλη λέγονται και class members.

Παράδειγμα

Το παρακάτω παράδειγμα δείχνει πως διαφορετικά αντικείμενα μοιράζονται την ίδια static μεταβλητή.

```
<?php
class Product
{
        static $fpa=23;

        function get_fpa()
        {
                //return self::$fpa;
                return static::$fpa;
```

```
        }
}

$milk = new Product();
$bread = new Product();

echo $milk->get_fpa(), " - ", $bread->get_fpa(), "<br />";

Product::$fpa=19;

echo $milk->get_fpa(), " - ", $bread->get_fpa();
?>
```

23 – 23
19 – 19

7.6 Σταθερές μεταβλητές (μεταβλητές με σταθερές τιμές)

Σε μία κλάση μπορούμε να δηλώσουμε μια μεταβλητή ως σταθερή με το λέξη: const.

```
class Product
{
  const FPA=0.23;
}
```

Όταν δηλώνεται μια μεταβλητή (έστω την FPA) ως σταθερή, δεν χρησιμοποιείται το $. Επίσης στην σταθερή πρέπει να δώσετε και αρχική τιμή. Μια σταθερή δεν μπορεί να είναι private ή public ή protected αλλά μόνο public εξ' ορισμού.

Για να έχετε πρόσβαση στην τιμή της μεταβλητής πρέπει να γραφτεί ως εξής:

Product::FPA;

Δηλαδή το όνομα της κλάσης, τον τελεστή :: και το όνομα της μεταβλητής.

Ισχύει δηλαδή ό,τι ισχύει και για τις static μεταβλητές.

Επίσης για να έχετε πρόσβαση στη μεταβλητή μέσα από μια συνάρτηση της κλάσης χρησιμοποιείτε πάλι (όπως και στις static) τη λέξη self::.

Δηλαδή γράφετε self::FPA και όχι $this->FPA.

```
<?php
class Product
{
```

```
  const FPA=0.23;
  function getFPA()
  {
    return self::FPA;
  }
}
echo Product::FPA;
$p1 = new Product();
echo $p1::FPA;
echo $p1->getFPA();
?>
```

Στο παραπάνω παράδειγμα και οι τρεις echo δίνουν το ίδιο αποτέλεσμα (0.23).

7.6.1 Υποκλάσεις

Μια υποκλάση κληρονομεί τις σταθερές της υπερκλάσης αλλά μπορεί επίσης να δημιουργήσει τις δικές της σταθερές ή και να υπερκαλύψει τις κληρονομημένες σταθερές.

```
<?php
class Milk extends Product
{
const FPA=0.19;
const SkimMilkFPA = 0.15;
  function get_parent_FPA()
  {
    return parent::FPA;
  }
}
echo Milk::FPA;
echo Milk::SkimMilkFPA;
$m = new Milk();
echo $m->getFPA();
echo $m->get_parent_FPA();
?>
```

Στο παραπάνω παράδειγμα τα τέσσερα echo θα μας έδιναν

0.19
0.15
0.23
0.23

7.7 Διεπαφή (interface)

7.7.1 Γενικές αρχές για διεπαφές

Μια διεπαφή είναι μια "κλάση" που περιέχει αποκλειστικά υπογραφές συναρτήσεων. Υπογραφή συνάρτησης λέγεται η συνάρτηση που δεν περιέχει σώμα (ούτε άγκιστρα).

Παράδειγμα

```
function getArea($width, $height);
```

Μια διεπαφή δηλώνεται με τον προσδιοριστή interface.

Παράδειγμα

```
interface Shape
{
  function getArea($width, $height);
}
```

- Μια διεπαφή μπορεί να περιέχει μεταβλητές οι οποίες είναι υποχρεωτικά const.
- Δεν μπορούμε να δημιουργήσουμε αντικείμενα από μια διεπαφή.
- Μια διεπαφή μπορεί να κάνει extend μία ή περισσότερες διεπαφές (πολυκληρονομηκότητα).
- Μια κλάση μπορεί να κάνει implement (υλοποίηση) μία ή περισσότερες διεπαφές.

Παράδειγμα

```
class Rec implements Shape
{
  function getArea($width, $height)
  {
    //υλοποίηση
  }
}
```

Μια κλάση που κάνει implement μια διεπαφή, πρέπει υποχρεωτικά να υλοποιεί όλες τις συναρτήσεις της διεπαφής.

Παράδειγμα

```php
<?php
interface Shape
{
        function getWidth();
        function getHeight();
        function setWidth($w);
        function setHeight($h);
        function getArea();
}

class Rectangle implements Shape
{
        private $width=0, $height=0;
        function getWidth()
        {
                return $this->width;
        }
        function getHeight()
        {
                return $this->height;
        }
        function setWidth($w)
        {
                $this->width=$w;
        }
        function setHeight($h)
        {
                $this->height=$h;
        }
        function getArea()
        {
                return $this->width*$this->height;
        }
}

$R = new Rectangle();
$R->setWidth(10);
$R->setHeight(5);
echo $R->getArea();
?>
```

Αποτέλεσμα εκτέλεσης

7.8 Αφηρημένη κλάση (abstract class)

7.8.1 Γενικές αρχές για αφηρημένες κλάσεις

Μια κλάση που περιέχει τουλάχιστον μία abstract συνάρτηση, δηλώνεται η ίδια ως abstract.

Μια συνάρτηση λέγεται abstract όταν δεν είναι υλοποιημένη ή αλλιώς όταν δεν έχει σώμα και άγγιστρα.

Μια αφηρημένη συνάρτηση δηλώνεται με τον προσδιοριστή abstract όπως παρακάτω:

```
abstract function draw();
```

Παράδειγμα

Η παρακάτω κλάση χαρακτηρίζεται ως abstract διότι περιέχει μια abstract συνάρτηση που είναι η draw();

```
abstract class GraphicObject
{
    private $x, $y;
    function moveTo($newX, $newY)
    {
        //υλοποίηση
    }
    abstract function  draw();
}
```

Δεν μπορούμε να δημιουργήσουμε αντικείμενα από μια αφηρημένη κλάση.

Μια αφηρημένη κλάση μπορεί να έχει τις δικές της μεταβλητές και συναρτήσεις.

Μια αφηρημένη κλάση μπορεί να έχει υποκλάσεις (μπορεί να γίνει extended).

(i) Αν η υποκλάση μιας αφηρημένης κλάσης υλοποιεί όλες τις αφηρημένες μεθόδους της υπερκλάσης, τότε είναι μια κανονική κλάση.

(ii) Αν η υποκλάση μιας αφηρημένης κλάσης δεν υλοποιεί όλες τις αφηρημένες μεθόδους της υπερκλάσης, ή έχει κάποια δική της αφηρημένη μέθοδο, τότε και η υποκλάση προσδιορίζεται ως abstract.

Μια αφηρημένη κλάση μπορεί να υλοποιήσει (implement) μια διεπαφή (interface).

Αν μια κλάση (όχι υποχρεωτικά αφηρημένη) κάνει implement μια διεπαφή και δεν υλοποιήσει έστω και μία μέθοδο της διεπαφής, τότε η κλάση προσδιορίζεται ως abstract.

Παράδειγμα

Στο παρακάτω παράδειγμα η κλάση Rectangle κάνει extend την GraphicObject η οποία είναι abstract (αφού περιέχει δύο abstract συναρτήσεις) και επομένως η Rectangle πρέπει να υλοποιήσει τις abstract της GraphicObject.

```
<?php
abstract class GraphicObject
{
    private $x, $y;
    function moveTo($newX, $newY)
    {
        //υλοποίηση
    }
        abstract function draw();  //αφηρημένη μέθοδος
        abstract function resize();  //αφηρημένη μέθοδος
}

class Rectangle extends GraphicObject
{
        function draw()
        {
                //υλοποίηση
        }
        function resize()
        {
                //υλοποίηση
        }
}
?>
```

8 Βάσεις Δεδομένων

8.1 Σύνδεση και διαχείριση Βάσης Δεδομένων

Η PHP από τις πρώτες εκδόσεις της έγινε πολύ δημοφιλής εξαιτίας της πολύ εύκολης συνδεσιμότητας με ένα επίσης ανοικτού κώδικα σύστημα διαχείρισης βάσεων δεδομένων, την MySQL. Η πρώτη επέκταση που υποστήριζε τη σύνδεση με μια βάση δεδομένων ήταν η επέκταση MySQL.

Η διαχείριση μιας βάσης δεδομένων απαιτεί αρχικά τη σύνδεση της PHP εφαρμογής με το Σύστημα Διαχείρισης Βάσεων Δεδομένων ΣΔΒΔ (όπως είναι η MySQL) το οποίο είναι εγκατεστημένο σε έναν Server ή DataBase Server. Σε κάθε Βάση Δεδομένων μπορεί να υπάρχουν ένας ή περισσότεροι πίνακες (tables) και σε κάθε πίνακα μπορεί να υπάρχουν καταχωρημένες μία ή περισσότερες εγγραφές (records).

Η εντολή σύνδεσης σε MySQL είναι:

```
mysql_connect(hostname, username, password);
```

hostname	Πρόκειται για τον server στον οποίο θα γίνει η σύνδεση. Η προκαθορισμένη τιμή είναι "localhost"
username	Πρόκειται για το username του χρήστη για σύνδεση στη MySQL.
password	Πρόκειται για το password του χρήστη για σύνδεση στη MySQL.

Στο παρακάτω παράδειγμα η εντολή mysql_connect() χρησιμοποιείται για τη σύνδεση σε μια MySQL στο localhost με username admin και password 12345 . Αν η σύνδεση αποτύχει επιστρέφεται ένα μήνυμα. Στη μεταβλητή $connection αποθηκεύει την σύνδεση.

```
<?php
$connection = mysql_connect("localhost","admin","12345");
if (!$connection)
{
 die('Η σύνδεση απέτυχε: ' . mysql_error());
}
?>
```

Αν η εντολή σύνδεσης αποτύχει εκτελείται η εντολή die που λειτουργεί σαν ένας συνδυασμός των echo + exit. Μετά το τέλος χρήσης της σλυνδεσης με την βάση δεδομένων είναι σωστή πρακτική να κλείνεται η σύνδεση χρησιμοποιώντας τη συνάρτηση mysql_close().

```
<?php
$ connection = mysql_connect("localhost"," admin ","12345");
```

```
if (!$connection)
{
 die('Η σύνδεση απέτυχε: ' . mysql_error());
}

mysql_close($connection);
?>
```

Μετά τη σύνδεση με τη MySQL, γίνεται σύνδεση με κάποια συγκεκριμένη βάση δεδομένων που είναι εγκατεστημένη στη MySQL. Όταν συνδεθεί με τη Βάση δεδομένων, η εφαρμογή μπορεί να εκτελέσει διάφορα ερωτήματα SQL(queries) προς τη βάση με σκοπό να:

- Ανακτήσει δεδομένα
- Εισαγει δεδομένα
- Τροποποιήσει δεδομένα
- Διαγράψει δεδομένα
- Ανάκτηση δεδομένων

Η ανάκτηση δεδομένων από έναν πίνακα μιας βάσης γίνεται με την εντολή **SELECT στη γλώσσα sql. Η** γλώσσα sql είναι case insesitive, δηλαδή δεν κάνει διάκριση ανάμεσα σε κεφαλαία και πεζά γράμματα.

Ας υποθέσουμε ότι στη βάση δεδομένων υπάρχει ο παρακάτω πίνακας με το όνομα "customers" ο οποίος έχει 4 πεδία (fields) με τα ονόματα LName, FName, Phone και Address και 4 εγγραφές (records).

LName	FName	**Phone**	Address
Joshua	Steve	2120303	NY
Andrew	Eve	254887	Boston
Smith	Bob	369875	Chicago
Tender	Mike	154781	Toronto

Το παρακάτω παράδειγμα επιλέγει όλες τις εγγραφές από τον πίνακα customers (ο χαρακτήρας * σημαίνει επιλογή όλων):

```
<?php
$conn = mysql_connect("localhost","admin","12345"); //σύνδεση με την
mysql
if (!$conn)
{
 die('Η σύνδεση απέτυχε: ' . mysql_error());
}

mysql_select_db("mydatabase", $conn); //επιλογή της ΒΔ (mydatabase)
```

```
$result = mysql_query("SELECT * FROM customers "); //επιλογή όλων των
εγγραφών από τον πίνακα persons

while($row = mysql_fetch_array($result)) //μετατροπή των στοιχείων
εγγραφής σε ένα Array
{
 echo $row['FName'] . " " . $row['LName']; //εμφάνιση των στοιχείων
εγγραφής
 echo "<br />"; //αναδίπλωση γραμμής
}

mysql_close($conn); //κλείσιμο της σύνδεσης
?>
```

Αντί για $row['FName'], $row['LName'] κλπ, μπορείτε να χρησιμοποιήσετε $row[0], $row[1], ... κλπ.

8.2 Εισαγωγή δεδομένων

Το παρακάτω παράδειγμα εισάγει μία εγγραφή στον πίνακα customers με την εντολή **INSERT INTO**

```
<?php
$conn = mysql_connect("localhost","admin","12345"); //σύνδεση με την
mysql
if (!$conn)
{
 die('Η σύνδεση απέτυχε: ' . mysql_error());
}

mysql_select_db("mydatabase", $conn); //επιλογή της ΒΔ (mydatabase)

$result = mysql_query("INSERT INTO customers VALUES ('Bill', 'Suzan',
'
 123444', 'NY')");
if (!$result)
{
 die('Η εγγραφή απέτυχε: ' . mysql_error());
}

mysql_close($conn); //κλείσιμο της σύνδεσης
?>
```

Μετά την εγγραφή ο πίνακας customers στη ΒΔ **mydatabase** θα δείχνει έτσι:

LName	FName	**Phone**	Address
Joshua	Steve	2120303	NY
Andrew	Eve	254887	Boston
Smith	Bob	369875	Chicago
Tender	Mike	154781	Toronto
Bill	Suzan	123444	NY

8.3 Ενημέρωση δεδομένων

Το παρακάτω παράδειγμα ενημερώνει μία εγγραφή στον πίνακα customers με την εντολή **UPDATE**

```
<?php
$conn = mysql_connect("localhost","admin","12345"); //σύνδεση με την
mysql
if (!$conn)
{
 die('Η σύνδεση απέτυχε: ' . mysql_error());
}

mysql_select_db("mydatabase", $conn); //επιλογή της ΒΔ (mydatabase)

$result = mysql_query("UPDATE customers SET Address = 'NY' WHERE
LName = 'Smith'");
if (!$result)
{
 die('Η ενημέρωση απέτυχε: ' . mysql_error());
}

mysql_close($conn); //κλείσιμο της σύνδεσης
?>
```

Μετά την εγγραφή ο πίνακας customers στη ΒΔ **mydatabase** θα δείχνει έτσι:

LName	FName	**Phone**	Address
Joshua	Steve	2120303	NY

Andrew	Eve	254887	Boston
Smith	Bob	369875	NY
Tender	Mike	154781	Toronto
Bill	Suzan	123444	NY

8.4 Διαγραφή δεδομένων

Για τη διαγραφή δεδομένων χρησιμοποιείτε την εντολή DELETE FROM. Το παρακάτω παράδειγμα διαγράφει μία εγγραφή από τον πίνακα customers με την εντολή **DELETE FROM**

```
<?php
$conn = mysql_connect("localhost","admin","12345"); //σύνδεση με την
mysql
if (!$conn)
{
 die('Η σύνδεση απέτυχε: ' . mysql_error());
}

mysql_select_db("mydatabase", $conn); //επιλογή της ΒΔ (mydatabase)

$result = mysql_query("DELETE FROM customers WHERE LName = 'Smith'
");
if (!$result)
{
 die('Η διαγραφή απέτυχε: ' . mysql_error());
}

mysql_close($conn); //κλείσιμο της σύνδεσης
?>
```

Μετά τη διαγραφή ο πίνακας customers στη ΒΔ **mydatabase** θα δείχνει έτσι:

LName	FName	**Phone**	Address
Joshua	Steve	2120303	NY
Andrew	Eve	254887	Boston
Tender	Mike	154781	Toronto
Bill	Suzan	123444	NY

Μετά την έκδοση PHP 5 η σύνδεση με την MySQL γίνεται εναλλακτικά με τη χρήση δύο επεκτάσεων οι οποίες εξασφαλίζουν τη δυνατότητα για ασφαλέστερη σύνδεση και διαχείριση των βάσεων δεδομένων:

- **Επέκταση MySQLi** (το "i" σημαίνει «improved» δηλαδή βελτιωμένη)
- **PDO (PHP Data Objects)**

Οι παλαιότερες εκδόσεις χρησιμοποιούσαν την επέκταση MySQL. Η επέκταση όμως αυτή αν και κάλυπτε την ανάγκη για εύκολη και άμεση σύνδεση του κώδικα PHP με τη βάση δεδομένων MySQL, δεν μπορούσε να ικανοποιήσει τις απαραίτητες προδιαγραφές ασφαλείας που προέκυπταν όταν κάποια κακόβουλη επίθεση εκδηλωνόταν. όπως πχ SQL query injection. Για το λόγο αυτό δημιουργήθηκαν οι επεκτάσεις MySQLi και PDO. Απαντώντας στο ερώτημα ποια από τις δύο επεκτάσεις είναι η καταλληλότερη, η απάντηση είναι ότι και οι δύο έχουν τα δικά τους πλεονεκτήματα:

- Η PDO υποστηρίζει πολλά διαφορετικά συστήματα διαχείρισης βάσεων δεδομένων ενώ η MySQLi υποστηρίζει μόνο την MySQL.
- Επίσης και οι δύο υποστηρίζουν αντικειμενοστραφή προγραμματισμό, όμως η MySQLi προσφέρει και ένα εναλλακτικό API για διαδικασιακό προγραμματισμό που διευκολύνει πολύ τους παλαιότερους προγραμματιστές που χρησιμοποιούσαν επί χρόνια την επέκταση MySQL.

Συνεπώς αν πρόκειται να γραφεί κάποια εφαρμογή η οποία ενδέχεται να αλλάξει μελλοντικά σύστημα διαχείρισης βάσης δεδομένων τότε είναι προτιμότερο να χρησιμοποιηθεί η επέκταση PDO ενώ με την MySQLi θα πρέπει να γραφεί ξανά μεγάλο μέρος του κώδικα.

Στα επόμενα παραδείγματα θα γίνει παρουσίαση των επεκτάσεων MySQLi και PDO με όλους τους πιθανούς τρόπους που μπορούν να χρησιμοποιηθούν , δηλαδή είτε χρησιμοποιώντας διαδικασιακό ή αντικειμενοσταραφή προγραμματισμό:

- MySQLi (αντικειμενοσταρφής προσέγγιση)
- MySQLi (διαδικασιακή προσέγγιση)
- PDO

8.5 Άνοιγμα σύνδεσης με βάση δεδομένων MySQL

LName	FName	**Phone**	Address
Joshua	Steve	2120303	NY
Andrew	Eve	254887	Boston

Smith	Bob	369875	Chicago
Tender	Mike	154781	Toronto

Για την εγκατάσταση μιας σύνδεσης με μια βάση της MySQL, μπορεί να χρησιμοποιηθεί εναλλακτικά ο διαδικασιακός και αντικειμενοστραφής τρόπος της MySQLi καθώς και η επέκταση PDO. Στα παραδείγματα παρακάτω εμφανίζονται και οι τρεις τρόποι.

MySQLi (Αντικειμενοστραφής μέθοδος)

```
<?php
$servername = "localhost";
$username = "admin";
$password = "12345";

// Create connection
$conn = new mysqli($servername, $username, $password);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?>
```

MySQLi (Με συνάρτηση)

```
<?php
$servername = "localhost";
$username = "admin";
$password = "12345";

// Create connection
$conn = mysqli_connect($servername, $username, $password);

// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
echo "Connected successfully";
?>
```

PDO

```
<?php
$servername = "localhost";
$username = "admin";
$password = "12345";

try {
    $conn = new PDO("mysql:host=$servername;dbname=myDatabase",
$username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    echo "Connected successfully";
    }
catch(PDOException $e)
    {
    echo "Connection failed: " . $e->getMessage();
    }
?>
```

Όπως παρατηρείτε στην περίπτωση σύνδεσης με PDO θα πρέπει κατά τη σύνδεση με τον διακομιστή της βάσης να προσδιορισθεί το όνομα της βάσης (“myDatabase”). Επίσης ένα μεγάλο πλεονέκτημα της PDO είναι ότι μπορεί να γίνει διαχείριση κάποιου σφάλματος στη σύνδεση με την εντολή try – catch μέσω της οποίας μπορεί να εντοπισθεί και να γίνει διαχείριση ενός exception.

Όπως έχουμε αναφέρει, μετά τη χρήση της βάσης θα πρέπει να γίνει κλείσιμο της σύνδεσης για εξοικονόμηση πόρων.

MySQLi (Αντικειμενοστραφής μέθοδος)

```
$conn->close();
```

MySQLi (Με συνάρτηση)

```
mysqli_close($conn);
```

PDO

```
$conn = null;
```

8.6 Προκαθορισμένα ερωτήματα SQL (Prepared Statements)

Όπως αναφέραμε η επέκταση MySQL της PHP που χρησιμοποιούνταν στις πρώτες εκδόσεις είχε πολλές αδυναμίες κυρίως σε θέματα ασφάλειας με αποτέλεσμα να είναι ευάλωτη σε επιθέσεις τύπου SQL injection. Το πρόβλημα αυτό αντιμετωπίστηκε στις επεκτάσεις MySQLi και PDO χρησιμοποιώντας τις προκαθορισμένες προτάσεις

(prepared statement). Πρόκειται για μια τεχνική που χρησιμοποιείται για την επαναλαμβανόμενη και ασφαλή εκτέλεση ερωτημάτων προς τη βαση δεδομένων με αποδοτικότερο τρόπο.
Η τεχνική αυτή λειτουργεί ως εξής:

Αρχικά δημιουργείται μια προκαθορισμένη εντολή SQL προς τη βάση δεδομένων αφήνοντας συγκεκριμένες παραμέτρους του ερωτήματος χωρίς τιμή και συμβολίζοντάς τες με τον χαρακτήρα "?") για παράδειγμα:

INSERT INTO tblusers VALUES(?, ?, ?)

Το σύστημα διαχείρισης της βάσης δεδομένων αποκωδικοποιεί και μεταφράζει το ερώτημα της SQL και αποθηκεύει το αποτέλεσμα της μετάφρασης χωρίς να το εκτελέσει.

Όταν αργότερα πρόκειται η PHP να εκτελέσει το ερώτημα, αποστέλλει τις τιμές που αντιστοιχούν στις παραμέτρους και η βάση δεδομένων εκτελεί το ερώτημα για τις συγκεκριμένες τιμές επιστρέφοντας τα αποτελέσματα. Η εφαρμογή σε PHP μπορεί να εκτελέσει το ίδιο προκαθορισμένο ερώτημα αρκετές φορές με διαφορετικές τιμές στις παραμέτρους.

Συγκρίνοντας την άμεση εκτέλεση των ερωτημάτων σε σχέση με τα προκαθορισμένα ερωτήματα, η τεχνική των prepared statements παρουσιάζει κάποια βασικά πλεονεκτήματα:

- Τα προκαθορισμένα ερωτήματα μειώνουν το χρόνο μετάφρασης και ως εκ τούτου το χρόνο εκτέλεσης τους ερωτήματος διότι το ερώτημα μεταφράζεται από το σύστημα διαχείρισ της βάσης δεδομένων μια φορά και παραμετροποιείται κάθε φορά που αποστέλλονται τιμές για τις παραμέτρους.
- Βοηθούν στην εξοικονόμηση πόρων για τον server διότι σε κάθε κλήση ενός ερωτήματος SQL αποστέλλονται μόνο οι παράμετροι και όχι ολόκληρο το ερώτημα εκ νέου.
- Τα prepared statements επιλύουν ένα βασικό πρόβλημα της ασφάλειας των δεδομένων που υφίστατο κατά τη χρήση των παλαιότερων επεκτάσεων σύνδεσης με την MySQL. Το πρόβλημα ασφάλειας σχετίζεται με μια από τις γνωστότερες επιθέσεις κατά βάσεων δεδομένων που είναι τα SQL injections (εμβόλιμες προτάσεις SQL). Το πρόβλημα αντιμετωπίζεται διότι οι τιμές των παραμέτρων περνούν αφού το ερώτημα σε SQL έχει ήδη μεταφραστεί και δεν μπορεί να παραποιηθεί.

8.6.1 Προκαθορισμένα ερωτήματα σε MySQLi

Στα επόμενα παραδείγματα γίνεται χρήση των προκαθορισμένων ερωτημάτων και με τους 2 τρόπους της MySQLi και της επέκτασης PDO.

MySQLi Prepared Statements

```
<?php
$servername = "localhost";
$username = "admin";
$password = "12345";
$dbname = "myDatabase";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// prepare and bind
$stmt = $conn->prepare("INSERT INTO customers (Fname, Lname, phone)
VALUES (?, ?, ?)");
$stmt->bind_param("sss", $firstname, $lastname, $phone);

// set parameters and execute
$firstname = " Steve ";
$lastname = " Joshua ";
$ phone = "2546998";
$stmt->execute();

$firstname = " Eve ";
$lastname = " Andrew ";
$email = "245787";
$stmt->execute();

echo "New records created successfully";

$stmt->close();
$conn->close();
?>
```

Το ερώτημα που πρόκειται να εκτελεστεί αφορά την εισαγωγή εγγραφών στον πίνακα customers αλλά οι τιμές των πεδίων δεν περνούν άμεσα αλλά χρησιμοποιούνται παράμετροι.

"INSERT INTO customers (fname, lname, phone) VALUES (?, ?, ?)"

Στην SQL, το σύμβολο ? αντικαθιστά τις τιμές που πρόκειται να περάσουν είτε είναι ακέραιου τύπου (integer), ή συμβολοσειρά (string), η πραγματικός (double) ή αντικειμένου blob.

Στη συνέχεια χρησιμοποιείται η συνάρτηση bind_param():
$stmt->bind_param("sss", $firstname, $lastname, $email);

Η συνάρτηση αυτή συνδέει τις παραμέτρους του ερωτήματος SQL και δηλώνει στη βάση δεδομένων τον τύπο της κάθε παραμέτρου. Επειδή στην προκειμένη περίπτωση και οι τρεις παράμετροι αντικαθιστούν τιμές τύπου string χρησιμοποιείται το όρισμα "sss" που αναπαριστά τα τρία string. Έτσι για κάθε παράμετρο που χρησιμοποιείται στο ερώτημα αντιστοιχεί και ένα σύμβολο ανάλογα με τον τύπο δεδομένου.

Έτσι τα σύμβολα είναι τα εξής:

- i – integer (ακέραιος)
- d – double (πραγματικός)
- s – string (συμβολοσειρά)
- b - BLOB

8.6.2 Prepared Statements σε PDO

Το ακόλουθο παράδειγμα χρησιμοποιεί prepared statements σε PDO:

PDO με Prepared Statements)

```
<?php
$servername = "localhost";
$username = "admin";
$password = "12345";
$dbname = "myDatabase";

try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname",
$username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // prepare sql and bind parameters
    $stmt = $conn->prepare("INSERT INTO customers (Fname, Lname,
phone)
```

```
    VALUES (:Fname, :Lname, :phone)");
    $stmt->bindParam(':Fname', $firstname);
    $stmt->bindParam(':Lname', $lastname);
    $stmt->bindParam(':phone', $email);

// set parameters and execute
$firstname = " Steve ";
$lastname = " Joshua ";
$ phone = "2546998";
$stmt->execute();

$firstname = " Eve ";
$lastname = " Andrew ";
$email = "245787";
$stmt->execute();



    echo "New records created successfully";
    }
catch(PDOException $e)
    {
    echo "Error: " . $e->getMessage();
    }
$conn = null;
?>
```

8.7 Ερώτημα Select στη βάση με MySQLi

Ένα ερώτημα σε SQL χρησιμοποιείται για να ανακτήσει τα δεδομένα από τον πίνακα.

SELECT * FROM table_name

MySQLi (Αντικειμενοστραφής)

```
<?php
$servername = "localhost";
$username = "admin";
$password = "12345";
$dbname = "myDatabase";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
```

```
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT id, fname, lname FROM customers ";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "id: " . $row["id"]. " - Name: " . $row["fname"]. " " .
$row["lname"]. "<br>";
    }
} else {
    echo "0 results";
}
$conn->close();
?>
```

Αρχικά ορίζεται ένα SQL query που ανακτά τα πεδία id, fname και lname από τον πίνακα customers. Η επόμενη εντολή εκτελεί το query και βάζει το αποτέλεσμα σε μια μεταβλητή που ονομάζεται $result.

Η συνάρτηση num_rows() ελέγχει αν έχουν επιστραφεί περισσότερα του ενός αποτελέσματα. Αν έχουν επιστραφεί αποτελέσματα, η συνάρτηση fetch_assoc() τοποθετεί όλα τα αποτελέσματα σε έναν συσχετιζόμενο πίνακα τον οποίο και μπορεί να προσπελάσει. Μια δομή επανάληψης while() διατρέχει όλα τα στοιχεία του πίνακα για να εμφανίσει τα αποτελέσματα.

MySQLi (Με συνάρτηση)

```
<?php
$servername = "localhost";
$username = "admin";
$password = "12345";
$dbname = "myDatabase";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
```

```
$sql = "SELECT id, fname, lname FROM customers ";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
    // output data of each row
    while($row = mysqli_fetch_assoc($result)) {
        echo "id: " . $row["id"]. " - Name: " . $row["fname"]. " " .
$row["lname"]. "<br>";
    }
} else {
    echo "0 results";
}

mysqli_close($conn);
?>
```

PDO

```
<?php
$servername = "localhost";
$username = "admin";
$password = "12345";
$dbname = "myDatabase";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = " SELECT id, fname, lname FROM customers ";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    echo "<table><tr><th>ID</th><th>Name</th></tr>";
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "<tr><td>".$row["id"]."</td><td>".$row["fname"]."
".$row["lname"]."</td></tr>";
    }
    echo "</table>";
} else {
    echo "0 results";
```

```
}
$conn->close();
?>
```

8.7.1 Select με PDO και Prepared Statements

Το παρακάτω παράδειγμα χρησιμοποιεί prepared statements για να εμφανίσει τα πεδία id, fname και lname από τον πίνακα customers μέσα σε έναν πίνακα HTML:

PDO

```
<?php
echo "<table style='border: solid 1px black;'>";
echo "<tr><th>Id</th><th>Firstname</th><th>Lastname</th></tr>";

class TableRows extends RecursiveIteratorIterator {
    function __construct($it) {
        parent::__construct($it, self::LEAVES_ONLY);
    }

    function current() {
        return "<td style='width:150px;border:1px solid black;'>" .
parent::current(). "</td>";
    }

    function beginChildren() {
        echo "<tr>";
    }

    function endChildren() {
        echo "</tr>" . "\n";
    }
}

$servername = "localhost";
$username = "admin";
$password = "12345";
$dbname = "myDatabase";

try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname",
$username, $password);
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $stmt = $conn->prepare("SELECT id, fname, lname FROM customers
```

```
");
    $stmt->execute();

    // set the resulting array to associative
    $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
    foreach(new TableRows(new RecursiveArrayIterator($stmt-
>fetchAll())) as $k=>$v) {
        echo $v;
    }
}
catch(PDOException $e) {
    echo "Error: " . $e->getMessage();
}
$conn = null;
echo "</table>";
?>
```

8.8 Εντολή Insert με MySQLi και PDO

Όταν μια βάση δεδομένων και ένας πίνακας έχουν δημιουργηθεί, μπορούν να προστεθούν δεδομένα σε αυτά. Αυτό μπορεί να γίνει με κάποιους κανόνες:

- Το SQL query πρέπει να είναι μέσα σε εισαγωγικά στην PHP
- Οι τιμές τύπου String μέσα σε ένα SQL query πρέπει να είναι μέσα σε εισαγωγικά
- Οι αριθμητικές τιμές καθώς και η NULL δεν πρέπει να μπαίνουν σε εισαγωγικά.

Για την εισαγωγή δεδομένων και εγγραφών σε έναν πίνακα της MySQL χρησιμοποιείται η εντολή INSERT INTO με σύνταξη:

INSERT INTO table_name (column1, column2, column3,...) VALUES (value1, value2, value3,...)

Αν ένα πεδίο είναι τύπου AUTO_INCREMENT (όπως συνήθως πεδία "id") ή τύπου TIMESTAMP τότε δεν είναι υποχρεωτικό να προσδιορίζονται στο SQL query διότι η MySQL θα προσθέσει αυτόματα την τιμή.

MySQLi (Αντικειμενοστραφής)

```
<?php
$servername = "localhost";
$username = "admin";
$password = "12345";
```

```
$dbname = "myDatabase";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "INSERT INTO customers (fname, lname, phone)
VALUES ('John', 'Pappas', 2457889')";

if ($conn->query($sql) === TRUE) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}

$conn->close();
?>
```

MySQLi (Με συνάρτηση)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "mydatabase";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

$sql = " INTO customers (Fname, Lname, phone) VALUES ('John',
‘pappas’, '2457888')";

if (mysqli_query($conn, $sql)) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}
```

```
mysqli_close($conn);
?>
```

PDO

```
<?php
$servername = "localhost";
$username = "admin";
$password = "12345";
$dbname = "myDatabase";

try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname",
$username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $sql = " INSERT INTO customers (Fname, Lname, phone)
    VALUES ('John', 'Pappas', '215447')";
    // use exec() because no results are returned
    $conn->exec($sql);
    echo "New record created successfully";
    }
catch(PDOException $e)
    {
    echo $sql . "<br>" . $e->getMessage();
    }

$conn = null;
?>
```

8.9 Εντολή Delete σε πίνακα με MySQLi και PDO

Η εντολή DELETE χρησιμοποιείται για τη διαγραφή τιμών από τον πίνακα και συντάσσεται ως εξής:

DELETE FROM table_name
WHERE some_column = some_value

Η εντολή WHERE χρησιμοποιείται για να προσδιορίσει ποιες εγγραφές θα διαγραφούν με βάση κάποια συνθήκη. Αν δεν συμπληρωθεί τότε θα διαγραφούν όλες οι εγγραφές.

Ας θεωρήσουμε τον παρακάτω πίνακα Customers

Id	LName	FName	**Phone**	Address
1	Joshua	Steve	2120303	NY
2	Andrew	Eve	254887	Boston
3	Smith	Bob	369875	Chicago
4	Tender	Mike	154781	Toronto

Το ακόλουθο παράδειγμα διαγράφει την εγγραφή με id=3 στον πίνακα customers:

MySQLi (Αντικειμενοστραφής)

```
<?php
$servername = "localhost";
$username = "admin";
$password = "12345";
$dbname = "myDatabase";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// sql to delete a record
$sql = "DELETE FROM customers WHERE id=3";

if ($conn->query($sql) === TRUE) {
    echo "Record deleted successfully";
} else {
    echo "Error deleting record: " . $conn->error;
}

$conn->close();
?>
```

MySQLi (Με συνάρτηση)

```
<?php
$servername = "localhost";
$username = "admin";
$password = "12345";
$dbname = "myDatabase";
// Create connection
```

```
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

// sql to delete a record
$sql = "DELETE FROM customers WHERE id=3";

if (mysqli_query($conn, $sql)) {
    echo "Record deleted successfully";
} else {
    echo "Error deleting record: " . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

PDO

```
<?php
$servername = "localhost";
$username = "admin";
$password = "12345";
$dbname = "myDatabase";

try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname",
$username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // sql to delete a record
    $sql = "DELETE FROM customers WHERE id=3";

    // use exec() because no results are returned
    $conn->exec($sql);
    echo "Record deleted successfully";
    }
catch(PDOException $e)
    {
    echo $sql . "<br>" . $e->getMessage();
    }
```

```
$conn = null;
?>
```

Μετά τη διαγραφή της εγγραφής ο πίνακας θα περιέχει τα ακόλουθα:

Id	LName	FName	**Phone**	Address
1	Joshua	Steve	2120303	NY
2	Andrew	Eve	254887	Boston
4	Tender	Mike	154781	Toronto

8.10 Εντολή Update με MySQLi και PDO

Η εντολή Update χρησιμοποιείται για την τροποποίηση ήδη υπάρχουσων εγγραφών στον πίνακα.

UPDATE table_name
SET column1=value, column2=value2,...
WHERE some_column=some_value

Η εντολή WHERE χρησιμοποιείται για να προσδιορίσει ποιες εγγραφές θα τροποποιηθούν με βάση κάποια συνθήκη. Αν δεν συμπληρωθεί τότε θα τροποποιηθούν όλες οι εγγραφές.

Μετά τη τροποποίηση της εγγραφής ο πίνακας θα περιέχει τα ακόλουθα:

Id	LName	FName	**Phone**	Address
1	Joshua	Steve	2120303	NY
2	Andrew	Eve	254887	Boston
4	Tender	Mike	154781	Toronto

Το ακόλουθο παράδειγμα δείχνει την τροποποίηση της εγγραφής με id=2:

MySQLi (Αντικειμενοστραφής)

```
<?php
$servername = "localhost";
$username = "admin";
```

```
$password = "12345";
$dbname = "myDatabase";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "UPDATE customers SET lname='Doe' WHERE id=2";

if ($conn->query($sql) === TRUE) {
    echo "Record updated successfully";
} else {
    echo "Error updating record: " . $conn->error;
}

$conn->close();
?>
```

MySQLi (Με συνάρτηση)

```
<?php
$servername = "localhost";
$username = "admin";
$password = "12345";
$dbname = "myDatabase";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

$sql = "UPDATE customers SET lname='Doe' WHERE id=2";

if (mysqli_query($conn, $sql)) {
    echo "Record updated successfully";
} else {
    echo "Error updating record: " . mysqli_error($conn);
}
```

```
mysqli_close($conn);
?>
```

PDO

```
<?php
$servername = "localhost";
$username = "admin";
$password = "12345";
$dbname = "myDatabase";

try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname",
$username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    $sql = "UPDATE customers SET lname='Doe' WHERE id=2";

    // Prepare statement
    $stmt = $conn->prepare($sql);

    // execute the query
    $stmt->execute();

    // echo a message to say the UPDATE succeeded
    echo $stmt->rowCount() . " records UPDATED successfully";
    }
catch(PDOException $e)
    {
    echo $sql . "<br>" . $e->getMessage();
    }

$conn = null;
?>
```

Μετά τη διαγραφή της εγγραφής ο πίνακας θα περιέχει τα ακόλουθα:

Id	LName	FName	**Phone**	Address
1	Joshua	Steve	2120303	NY
2	Doe	Eve	254887	Boston
4	Tender	Mike	154781	Toronto

8.1 Εργαλεία εκτέλεσης κώδικα PHP

Η ανάπτυξη του κώδικα μπορεί να γίνει με τον οποιονδήποτε συντάκτη κειμένου. Ένα πολύ καλό εργαλείο για τη σύνταξη κώδικα σκριπτ είναι το Notepad++. Η δοκιμή και μεταγλώττιση του κώδικα PHP μπορεί να γίνει με το λογισμικό ανοικτού κώδικα WAMP (Windows – MySQL – PHP – Apache). Το WAMP είναι μια συλλογή πολλών εργαλείων. Το όνομά του προέρχεται από τα αρχικά του λειτουργικού συστήματος των Microsoft Windows και τα συστατικά Apache, MySQL και PHP. Η MySQL είναι σύστημα διαχείρισης βάσεων δεδομένων το οποίο σε συνδυασμό με τη γλώσσα PHP είναι μια από τις δημοφιλέστερες πλατφόρμες ανάπτυξης δυναμικών ιστοσελίδων. Επίσης μπορεί να χρησιμοποιηθεί το phpMyAdmin το οποίο παρέχει ένα γραφικό περιβάλλον στο διαχειριστή της βάσης δεδομένων MySQL. Η

Ο Apache είναι ένας πολύ ισχυρός και από τους πιο διαδεδομένους web server Ο Apache HTTP server είναι ένας web server που έχει ως βασικό ρόλο την παροχή του περιεχομένου των ιστοσελίδων στις εφαρμογές πελάτης όπως είναι οι web browsers. Αποτελεί το πλέον διαδεδομένο εξυπηρετητή σε συστήματα Unix από την άποψη της λειτουργικότητας και απόδοσης. Μέχρι πριν την ύπαρξη του WAMP ο Apache λειτουργούσε μόνο κάτω από ένα λειτουργικό σύστημα τύπου UNIX. Το WAMP έδωσε τη δυνατότητα στους προγραμματιστές να χρησιμοποιούν μια έκδοση του Apache σε Windows. Η εφαρμογή είναι πλέον διαθέσιμη για μια μεγάλη ποικιλία λειτουργικών συστημάτων , συμπεριλαμβανομένων των Unix, το GNU , FreeBSD , Linux , Solaris , Mac OS X , Microsoft Windows.

Βιβλιογραφία

1. Edelstein, Herb. "Unraveling Client/Server Architecture." *DBMS 7,* 5 (May 1994)
2. http://php.net/
3. http://www.mysql.com/
4. Φεβρουάριος 2009 Web server Survey" . Netcraft . http://news.netcraft.com/archives/2009/02/18/february_2009_web_server_survey .html .
5. Terry Halpin, Tony Morgan, 2010, “Information Modeling and Relational Databases”, Morgan Kaufmann
6. Brett McLaughlin,2012, “PHP & MySQL: The Missing Manual”,"O'Reilly Media, Inc
7. http://www.w3schools.com/php/php_mysql_intro.asp

Παράρτημα

Ασκήσεις

Άσκηση 1

Εκκινήστε το WAMP και στον φάκελο C:/wamp/www δημιουργήστε με το Notepad το αρχείο helloworld.php. Στο αρχείο γράψτε κώδικα PHP που εμφανίζει το μήνυμα Hello ακολουθούμενο από το όνομά σας.

Λύση

```
<?php
echo("hello myname");
?>
```

Άσκηση 2

Στον φάκελο C:/wamp/www δημιουργήστε με το Notepad το αρχείο math.php. Στο αρχείο γράψτε κώδικα PHP που αναθέτει σε μεταβλητές x και y δύο ακέραιες τιμές και εμφανίζει το άθροισμα και τη διαφορά.

Λύση

```
<?php
$x = 59;
$y = 85;
$sum=$x+$y;
$dif=$x-$y;
echo("athroisma ".$sum.” diafora “.$dif);
?>
```

Άσκηση 3

Στον φάκελο C:/wamp/www δημιουργήστε με το Notepad το αρχείο func.php. Στο αρχείο γράψτε κώδικα PHP που αναθέτει σε μεταβλητές x και y δύο ακέραιες τιμές και με τη βοήθεια συναρτήσεων θα εμφανίζει το άθροισμα και τη διαφορά.

Λύση

Κώδικας	Αποτέλεσμα

<?php
function sum($x1,$y1) {
$s=$x1+$y1;
echo("athroisma ".$s);
}
 	
function dif($x1,$y1) {
$d=$x1+$y1;
echo("diafora ".$d);
}
 	
$x = 90;
$y = 85;
sum($x,$y);
dif($x,$y);
?>
	athroisma 175
	
	diafora 5

Άσκηση 4

Στον φάκελο C:/wamp/www δημιουργήστε με το Notepad το αρχείο choice.php. Στο αρχείο γράψτε κώδικα PHP που αναθέτει σε μεταβλητές x και y δύο ακέραιες τιμές και εμφανίζει το μικρότερο αριθμό.

Λύση

Κώδικας	Αποτέλεσμα

<?php
$x = 59;
$y = 85;
if($x<$y)
echo("mikrotero ".$x);
else
echo("mikrotero ".$y);
?>
	Mikrotero 59

Άσκηση 5

Στον φάκελο C:/wamp/www δημιουργήστε με το Notepad το αρχείο loop.php. Στο αρχείο γράψτε κώδικα PHP που εμφανίζει όλους τους τριψήφιους που είναι πολλαπλάσια του 8 και του 3.

Λύση

Κώδικας	Αποτέλεσμα

<?php
for ($i=100; $i<1000; $i++)
{
If(($i%8==0)&&($i%3==0))
echo $i . "
";
}
?>	

Άσκηση 6

Στον φάκελο C:/wamp/www δημιουργήστε με το Notepad το αρχείο loop2.php. Στο αρχείο γράψτε κώδικα PHP που εμφανίζει όλους τους τριψήφιους που είναι πολλαπλάσια του 8 και του 3 με τη χρήση της εντολής while.

Λύση

Κώδικας	Αποτέλεσμα

<?php
$i=100;
while($i<1000)
{
If(($i%8==0)&&($i%3==0))
echo $i . "
";
$i++;
}
?>	