# ΠΕΡΙΕΧΟΜΕΝΑ

# 5 Προγραμματισμός Εφαρμογών Που Εκτελούνται Στον Εξυπηρετητή

## 5.1 Εισαγωγή

Οι δυναμικές ιστοσελίδες, οι οποίες χρησιμοποιούν εφαρμογές που εκτελούνται στην πλευρά του εξυπηρετητή, α) λαμβάνουν τα δεδομένα που εισάγει ο χρήστης, β) τα επεξεργάζονται ή ανατρέχουν σε κάποια βάση δεδομένων που είναι εγκατεστημένη στον εξυπηρετητή, γ) δημιουργούν μία προσαρμοσμένη ιστοσελίδα, την οποία και δ) παρουσιάζουν τελικά στο χρήστης. Το περιεχόμενο της παραγόμενης ιστοσελίδας εξαρτάται από την ερμηνεία του προγραμματιστικού σεναρίου. Σε αντίθεση με την τεχνολογία που χρησιμοποιείται στον προγραμματισμό από την πλευρά του πελάτη όπως η JavaScript, ο προγραμματισμός από την πλευρά του εξυπηρετητή επιστρέφει προσαρμοσμένες ιστοσελίδες HTML. Οι ιστοσελίδες αυτές δεν περιέχουν άλλο κώδικα εκτός από τις ετικέτες HTML και μπορούν να εμφανίζονται σε όλους τους φυλλομετρητές, χωρίς να απαιτείται κάποια επιπλέον ρύθμιση σε αυτούς, όπως π.χ. ενεργοποίηση της εκτέλεσης σεναρίων Java.

Για την κατασκευή αυτών των προγραμμάτων μπορεί να χρησιμοποιηθούν διάφορες τεχνολογίες και γλώσσες σεναρίων. Οι πιο διαδεδομένες τεχνολογίες, τις οποίες αποτελούν αντικείμενο μελέτης του παρόντος κεφαλαίου, είναι οι: CGI (με PERL) και PHP.

Η **PHP** (Personal HomePage Tools) είναι μία γλώσσα και διερμηνέας σεναρίων που είναι ελεύθερα διαθέσιμη. Αρχικά χρησιμοποιήθηκε σε κεντρικούς υπολογιστές με περιβάλλον UNIX, αλλά μπορεί να χρησιμοποιηθεί και σε άλλα λειτουργικά συστήματα. Μία σελίδα HTML που περιλαμβάνει σενάρια PHP έχει τη χαρακτηριστική επέκταση ".php", ".php3", ".php4" ή ".phtml".

Η τεχνολογία **CGI** (Common Gateway Interface) είναι μία διαδεδομένη μέθοδο προγραμματισμού από την πλευρά του εξυπηρετητή. Αποτελεί ένα πρωτόκολλο επικοινωνίας ανάμεσα στους εξυπηρετητές του Ιστου (Apache, IIS κλπ) και στα διάφορα εκτελέσιμα αρχεία, προγράμματα, που είναι εγκατεστημένα στον εξυπηρετητή (server). Το προϊόν αυτής της επικοινωνίας είναι δυναμικά διαμορφωμένες ιστοσελίδες. Για την κατασκευή σεναρίων CGI μπορεί να χρησιμοποιηθούν πολλές γλώσσες γενικού προγραμματισμού, όπως η C, η C++ και η PERL, την οποία και θα μελετήσουμε. Η Perl (Practical Extraction και Report Language) είναι μία από τις πρώτες γλώσσες που χρησιμοποιήθηκαν για τον προγραμματισμό στον παγκόσμιο Ιστό. Ο αριθμός των έτοιμων βιβλιοθηκών προγραμμάτων που είναι διαθέσιμα αυτήν την στιγμή την καθιστούν ένα ευπροσάρμοστο εργαλείο. Πολλοί άνθρωποι βρίσκουν δύσκολο

τον προγραμματισμό σε Perl και αυτός είναι πιθανώς ο λόγος για τον οποίο έχουν γίνει δημοφιλέστερες άλλες, ευκολότερες και πιο εξειδικευμένες λύσεις. Εντούτοις, αποτελεί μια ολοκληρωμένη γλώσσα προγραμματισμού, η οποία μπορεί να χρησιμοποιηθεί και στην τεχνολογία CGI. Ο κώδικας της PERL δεν ενσωματώνεται σε αυτόν της σελίδας HTML, αλλά σε αυτόνομα αρχεία αποθηκευμένα στον εξυπηρετητή. Τα αρχεία αυτά καλούνται από το φυλλομετρητή του πελάτη, μέσω της ιστοσελίδας και εκτελούνται στον εξυπηρετητή. Το αποτέλεσμα της εκτέλεσης δεν είναι τίποτα άλλο παρά κώδικας HTML, ο οποίος αποστέλλεται στον πελάτη και εμφανίζεται ως ιστοσελίδα. Τα αρχεία που περιέχουν προγράμματα σε Perl έχουν συνήθως επέκταση ".pl".

Ο Πίνακας 5.1.1 παρουσιάζει μία απλή σύγκριση αυτών των τεχνολογιών.

Η σύγκριση των τεχνολογιών είναι δύσκολη. Πολλοί προγραμματιστές προτιμούν τη μέθοδο που τυχαίνει να ξέρουν καλύτερα ή τη μέθοδο που είναι ήδη καθιερωμένη στον οργανισμό στον οποίο εργάζονται ή σπουδάζουν. Η Perl μπορεί να απαιτεί περισσότερο χρόνο για την εκμάθησή της, σε σχέση με τις άλλες εναλλακτικές τεχνολογίες εντούτοις αποτελεί μία πλήρης γλώσσα προγραμματισμού για τον Ιστό και παρέχει μεγαλύτερη ευχέρεια υλοποίησης. Η PHP συνδυάζει μία ισχυρή γλώσσα συγγραφής σεναρίων, με σχετική ευκολία χρήσης, η οποία μπορεί να είναι η προφανής επιλογή για κάποιον αρχάριο προγραμματιστή, που χρειάζεται γρήγορα αποτελέσματα χωρίς πρόσθετες δαπάνες λογισμικού.

Πίνακας 5.1.1 – Σύγκριση τεχνολογιών Perl και PHP

| | **Perl** | **PHP** |
|---|---|---|
| Κόστος | Δωρεάν | Δωρεάν |
| Ενσωμάτωση σε σελίδα HTML | Όχι | Ναι |
| Ευκολία χρήσης | + | +++ |
| Διεπαφή με βάση δεδομένων | +++ | ++ |
| Χειρισμός κειμένου | +++ | +++ |
| Χειρισμός Εικόνων | ++ | ++ |

## 5.2 Η γλώσσα σεναρίων PHP

Η PHP είναι μία ισχυρή γλώσσα για τη συγγραφή σεναρίων που συνεργάζεται πολύ καλά με την HTML. Αποτελεί δε ένα πολύ καλό εργαλείο για το σχεδιασμό δυναμικών ιστοσελίδων.

Ένα από τα σημαντικότερα χαρακτηριστικά της PHP είναι ότι είναι μία γλώσσα ανοιχτού κώδικα και διανέμεται ελεύθερα, που σημαίνει ότι κάθε υλικό αναφοράς της μπορεί να εντοπιστεί εύκολα και να χρησιμοποιηθεί από τον καθένα. Επίσης μπορεί να χρησιμοποιηθεί με ευκολία σε διάφορα λειτουργικά συστήματα.

Για τον έλεγχο των προγραμμάτων PHP απαιτείται ένας εξυπηρετητής που υποστηρίζει τη γλώσσα.

### 5.2.1 Τι ακριβώς είναι η PHP

Η PHP είναι ένα πρόγραμμα που μπορεί να εγκατασταθεί σαν ενσωματωμένο επιπλέον λογισμικό (module) σε έναν εξυπηρετητή ιστού (Web server), είτε σαν αυτόνομος εκτελέσιμος κώδικας. Προτιμάται η πρώτη περίπτωση, αφού έτσι είναι γρηγορότερη η εκτέλεση και δεν επιβαρύνεται ο εξυπηρετητής με επιπλέον απαιτήσεις σε μνήμη και χώρο στον δίσκο. Χρησιμοποιείται με εκδόσεις των Apache, Microsoft IIS, Netscape Enterprise Server και άλλων πακέτων.

Η σύνταξη της γλώσσας PHP είναι παρόμοια με αυτήν της PERL και της C. Στην πραγματικότητα η PHP ξεκίνησε σαν μία εύκολη Perl και γράφτηκε τα τέλη του 1994 από τον Rasmus Lerdorf. Τα επόμενα χρόνια εξελίχθηκε στην PHP/FI 2.0. και PHP/FI. Το καλοκαίρι του 1997 οι Zeev Suraski και Andi Gutmans έφτιαξαν ένα καινούριο επεξεργαστή, οποίος οδήγησε στην PHP 3.0. Η έκδοση PHP 3.0. όρισε την σύνταξη για τις εκδόσεις 3 και 4.

Όταν κάποιος πελάτης (χρήστης) επισκεφτεί την ιστοσελίδα που περιέχει κώδικα PHP, ο εξυπηρετητής εκτελεί τον κώδικα και αυτό που επιστρέφεται στον χρήστη είναι μία ιστοσελίδα HTML. Όλη τη δουλειά την κάνει ο εξυπηρετητής και όχι κάποιος φυλλομετρητής. Ο πελάτης (χρήστης) από την πλευρά του δε χρειάζεται κάποιο πρόσθετο εργαλείο ή πρόγραμμα για να δει το αποτέλεσμα της PHP — φτάνει στον χρήστη ως κώδικας HTML.

Η PHP είναι μία γλώσσα για τη συγγραφή σεναρίων. Αυτό σημαίνει πως ο κώδικας, όπως και της HTML, δεν χρειάζεται να μεταγλωττιστεί πριν χρησιμοποιηθεί.

**Σημείωση**: Επισκεφτείτε τη σελίδα www.php.net Είναι ένα κέντρο ελέγχου, με πολύ υλικό αναφοράς, για τη γλώσσα και συμβουλές από χρήστες που χρησιμοποιούν και ασχολούνται με την PHP παγκόσμια.

Η PHP μπορεί να :

- Πάρει πληροφορία από φόρμες και να τη χρησιμοποιήσει με διάφορους τρόπους: να την αποθηκεύσει σε μία βάση δεδομένων, να δημιουργήσει εξαρτημένες/υποθετικές σελίδες που εξαρτώνται από τα περιεχόμενα της φόρμας, να τοποθετήσει cookies στο φυλλομετρητή του χρήστη και να στείλει e-mail.
- Πιστοποιήσει την αυθεντικότητα και να εντοπίσει χρήστες.
- Φιλοξενήσει συζητήσεις στην ιστοσελίδα που κατασκευάζουμε.
- Προσαρμόσει την εμφάνιση μιας ιστοσελίδας στους διαφορετικούς φυλλομετρητές ή συσκευές, που χρησιμοποιούν οι χρήστες.
- Δημοσιεύσει μία ολόκληρη ιστοσελίδα χρησιμοποιώντας μόνο μία φόρμα σχεδίου (server-side includes-style).
- Εξυπηρετήσει σελίδες XML.

### 5.2.2 Βασικοί κανόνες συγγραφής σεναρίων PHP

Πριν προχωρήσουμε παρακάτω στην μελάτη της PHP, θα χρειαστεί να ρίξουμε μία γρήγορη ματιά στα δομικά τμήματά της, ξεκινώντας από ένα παράδειγμα. Το παρακάτω πρόγραμμα περιέχεται στο αρχείο με όνομα "test.php". Όταν καλείται από έναν φυλλομετρητή, ο χρήστης απλά διαβάζει: "Αυτό είναι ένα παράδειγμα!".

```
<?php
  echo "Αυτό είναι ένα παράδειγμα!";
?>
```

Όλα τα σενάρια PHP περικλείονται στις εντολές: `<?php` και `?>`, που είναι οι ετικέτες αρχής και τέλους του προγράμματος PHP. Το παραπάνω πρόγραμμα χρησιμοποιεί την εντολή echo για να εκτυπώσει τη φράση στο "παράθυρο" του φυλλομετρητή.

Ο παραπάνω κώδικας γράφεται εντός της σελίδας HTML η οποία ονομάζεται `αρχείο1.php`, ως εξής:

```
<html>
<head><title> Παράδειγμα </title></head>
<body>
<font color="red">Ο PHP κώδικας δημιουργεί μια σελίδα
```

```
που λέει:</font>
<p>
  <?php
  print ("Αυτό είναι ένα παράδειγμα!");
  ?>
</body>
</html>
```

Η HTML διαχειρίζεται τον κώδικα ως απλή HTML, αλλά οτιδήποτε βρίσκεται μέσα στα `<?php` και `?>` θα τα επεξεργαστεί ως PHP.

***Σημείωση**: Υπάρχουν στο Διαδίκτυο διάφοροι ιστότοποι όπου μπορείτε εύκολα να δοκιμάσετε απλό κώδικα PHP. Ένας από αυτούς είναι και ο http://sandbox.onlinephpfunctions.com/, .οπού μπορείτε να 'δοκιμάσετε' τα παραδείγματα του βιβλίου.*

Οι βασικοί κανόνες συγγραφής της PHP παρουσιάζονται στις ακόλουθες παραγράφους.

## Ορισμός κώδικα

Κάθε τμήμα κώδικα PHP ξεκινάει με "<?php" (ή σύντομα "<?" αν το δέχεται ο εξυπηρετητής).

Τελειώνουμε τον κώδικα PHP γράφοντας "?>" στο τέλος του.

## Ονομασία αρχείων

Για να λειτουργήσει ένα πρόγραμμα σε PHP, θα πρέπει το αρχείο στο οποίο είναι ο κώδικας ή οποιοδήποτε αρχείο καλείται μέσα στον κώδικα να έχει την κατάληξη `.php` (παλιότερες εκδόσεις χρησιμοποιούσαν τις επεκτάσεις `.php3` και `.phtml`). Όπως και στην HTML, τα αρχεία αποθηκεύονται ως απλό κείμενο.

***Σημείωση**: Είναι δυνατόν ένα αρχείο να περιέχει κώδικά php αλλά να έχει διατηρήσει την επέκταση .html. Αυτό πραγματοποιείται με κατάλληλη διαφοροποίηση του αρχείου .htaccess του εξυπηρετητή Διαδικτύου.*

## Εισαγωγή σχολίων

Στην PHP κάθε μονή γραμμή σχολίου ξεκινάει με "//". Αν θέλουμε να γράψουμε ένα μπλοκ σχολίων ή να βγάλουμε προσωρινά εκτός επεξεργασίας κάποιο κομμάτι κώδικα μπορούμε να το βάλουμε εντός των συμβόλων "/*" και "*/":

```
<?php
// Αυτές οι γραμμές θα αγνοηθούν.
// Αποτελούν σχόλια
print ("Αυτό είναι ένα παράδειγμα!");
/*
και αυτές οι γραμμές θα αγνοηθούν. Αποτελούν ένα τμήμα
σχολίων.
*/
?>
```

**Διαχωρισμός εντολών**

Κάθε γραμμή εντολών τελειώνει με το σύμβολο ";".

Όπως και στην HTML, η μορφοποίηση του κώδικα της PHP (όπου βάζουμε κενά, αλλαγή γραμμής, κτλ.) δεν επηρεάζει το αποτέλεσμα. Έτσι ένα σύνολο από εντολές θα μπορούσαν να βρίσκονται όλες στην ίδια γραμμή, αρκεί να μην παραλείπονται τα ερωτηματικά. Εξαίρεση αποτελεί η μορφοποίηση που έχει εφαρμοστεί στα τμήματα του κώδικα που ορίζουν στο φυλλομετρητή πώς θα εμφανίσει την ιστοσελίδα.

Για παράδειγμα ο κώδικας:

```
<?php
     print ("Αυτό είναι ένα παράδειγμα!");
?>
```

θα έχει το ίδιο αποτέλεσμα στην οθόνη του χρήστη με τον παρακάτω:

```
<?php print ("Αυτό είναι ένα παράδειγμα!"); ?>
```

**Συναρτήσεις εξόδου**

Μέχρι στιγμής για την εκτύπωση κάποιου κειμένου στην οθόνη του φυλλομετρητή έχει χρησιμοποιηθεί η συνάρτηση "print".

Η τυπική συνάρτηση φαίνεται ως εξής:

```
print ( );
```

Η "print" είναι η συνάρτηση και τα δεδομένα που επεξεργάζεται βρίσκονται μέσα τις παρενθέσεις. Εναλλακτικά της "print" θα μπορούσε να χρησιμοποιηθεί η "echo". Οι δυο συναρτήσεις μοιάζουν σε αρκετά, καθώς και οι δυο αποτελούν δομικά στοιχεία της γλώσσας PHP και χρησιμοποιούνται ώστε να εκτυπώνουν δεδομένα στην οθόνη. Ενώ και οι δύο πρόκειται για συναρτήσεις, εντούτοις αποτελούν εξαίρεση καθώς λειτουργούν με ή χωρίς τις παρενθέσεις. Οπότε μπορεί να βρίσκονται με τη μορφή print () και echo () ή print και echo.

Η διαφορά τους έγκειται στον τρόπο λειτουργίας τους. Η print δέχεται μια παράμετρο ως δεδομένο εισόδου και επιστρέφει κάποια τιμή (που είναι πάντα ίση με 1). Ενώ η echo μπορεί να έχει περισσότερες της μιας παραμέτρους εισόδου και δεν επιστρέφει κάποια τιμή, το οποίο την καθιστά σχετικά γρηγορότερη. Αντίθετα, εξαιτίας της μη επιστροφής τιμής, η echo δε μπορεί να χρησιμοποιηθεί ως τμήμα πολύπλοκων διαδικασιών.

Το παρακάτω παράδειγμα παρουσιάζει τη χρήση της echo με δύο παραμέτρους εισόδου. Οι παράμετροι διαχωρίζονται με κόμμα.

```
echo "μπλα μπλα μπλα 1", "μπλα μπλα μπλα 2";
```

***Σημείωση**: Η παρακάτω γραμμή κώδικα αποτελεί παράδειγμα πολύπλοκων διαδικασιών, στις οποίες δε μπορεί να χρησιμοποιηθεί η συνάρτηση echo.*
*$b ? print "Είναι TRUE" : print "Είναι FALSE";*
*Ουσιαστικά αποτελεί μια ερώτηση για την τιμή της μεταβλητής $b. Εάν είναι αληθής (true) τότε θα εκτυπωθεί το πρώτο λεκτικό, δηλαδή η φράση "Είναι TRUE". Στην αντίθετη περίπτωση, δηλαδή η μεταβλητή $b είναι αναληθής (false), τότε θα εκτυπωθεί το δεύτερο λεκτικό.*

Μια λίγο διαφορετική συνάρτηση εξόδου αποτελεί η print_r. Η print_r δέχεται δυο παραμέτρους εισόδου. Η μια είναι η μεταβλητή για την οποία εκτυπώνει πληροφορίες, σε ανθρώπινα κατανοητό τρόπο. Αν η παράμετρος εισόδου της είναι απλή μεταβλητή, τότε εκτυπώνει την τιμή της. Διαφορετικά εάν πρόκειται για μεταβλητή πίνακα, τότε εκτυπώνει τη θέση και την τιμή του κάθε στοιχείου του. Η δεύτερη παράμετρος εισόδου της print_r καθορίζει αν η συνάρτηση θα επιστρέψει κάποια τιμή ή απλά θα κάνει εκτύπωση στην οθόνη του χρήστη. Παραδείγματα της χρήσης της print_r θα δούμε όταν εξετάσουμε τις μεταβλητές και κυρίως τις μεταβλητές πίνακα.

***Σημείωση**: Το παρακάτω εκτυπώνει την τιμή της μεταβλητής στην οθόνη του φυλλομετρητή: <?=$variable?>. Αποτελεί μια 'σύντομη εκδοχή' της echo.*

**Κλήσεις προγραμμάτων PHP**

Υπάρχουν δυο διαφορετικές προσεγγίσεις στην εισαγωγή/ κλήση κώδικα PHP μέσα από ένα αρχείο HTML.

Μικρά τμήματα κώδικα php μπορεί να ενσωματωθούν με τις ετικέτες HTML όπως παρουσιάζεται στο παρακάτω παράδειγμα.

```
<head></head>
<body>
Γεια, σήμερα είναι <?php echo date('l, d-m-Y'); ?>.
</body>
</html>
```

Το οποίο θα έχει ως αποτέλεσμα την εκτύπωση της φράσης «Γεια, σήμερα είναι Thursday, 15-09-2016».

Οτιδήποτε δεν περιέχεται εντός των συμβόλων του <?php και ?> αγνοείται από τον επεξεργαστή της PHP. Στο ανωτέρω παράδειγμα το τμήμα του κώδικά PHP χρησιμοποιεί τη συνάρτηση date ώστε να μορφοποιήσει και να εκτυπώσει την τοπική ημερομηνία στην οθόνη του χρήστη. Ο συγκεκριμένος ορισμός της date θα εκτυπώσει το πλήρες όνομα της μέρας, δυο ψηφία για την ημέρα, δύο ψηφία για το μήνα και τέσσερα ψηφία για το έτος.

Εάν τα τμήματα του κώδικα που πρέπει να ενσωματωθούν είναι αρκετά, και ίσως πολύπλοκα, τότε υπάρχει η δυνατότητα να αποδοθεί όλη η ιστοσελίδα με εντολές PHP. Έτσι το ανωτέρω παράδειγμα γίνεται:

```
<?php
  echo "<html>";
  echo "<head></head>";
  echo "<body>";
  echo "Γεια, σήμερα είναι ";
  echo date('l, d-m-Y');
  echo ".";
  echo "</body>";
  echo "</html>";
?>
```

Το αποτέλεσμα που θα επιστραφεί στο φυλλομετρητή του χρήστη θα είναι το κείμενο:
«<html><head></head><body> Γεια, σήμερα είναι Thursday, 15-09-2016.</body></html>»
Το οποίο πρόκειται, τελικά, για μια ιστοσελίδα που θα εμφανίσει στην οθόνη του φυλλομετρητή την φράση «Γεια, σήμερα είναι Thursday, 15-09-2016.».

Γενικά είναι αποδεκτό ότι ο πρώτος τρόπος είναι γρηγορότερος. Εντούτοις πολλοί υποστηρίζουν ότι ο κόπος της συντήρησης μιας ιστοσελίδας

που έχει εμβόλιμα αρκετά τμήματα κώδικα PHP αντισταθμίζει το πλεονέκτημα της ταχύτητας εκτέλεσης. Τελικά αποτελεί επιλογή του προγραμματιστή ποιόν τρόπο θα ακολουθήσει.

Ως πιο προχωρημένη τεχνική θεωρείται η χρήση των εντολών include και require για την εισαγωγή ολόκληρου αρχείου PHP. Έτσι, έστω ότι το αρχείο με όνομα test.php περιέχει τα παρακάτω:

```
<?php echo date('l, d-m-Y'); ?>
```

Έστω ότι χρειάζεται να το ενσωματώσουμε στην ιστοσελίδα, τότε θα πρέπει να γραφεί κάτι σαν το παρακάτω:

```
<head></head>
<body>
Γεια, σήμερα είναι <?php include 'test.php' ?>.
</body>
</html>
```

Ή εναλλακτικά

```
<head></head>
<body>
Γεια, σήμερα είναι <?php require 'test.php' ?>.
</body>
</html>
```

Αποτελούν εξαιρετικά χρήσιμα εργαλεία. Για παράδειγμα ο ορισμός ενός μενού πλοήγησης που επαναλαμβάνεται σε αρκετές ιστοσελίδες θα μπορούσε να ενσωματωθεί με έναν από αυτούς τους δύο τρόπους. Επιπλέον, έτσι μπορεί να ενσωματωθεί κώδικας που αριθμεί αρκετές γραμμές.
Η βασικότερη διαφορά ανάμεσα στην include και τη require αποτελεί το γεγονός ότι εάν το αρχείο (test.php) δε βρεθεί κατά την κλήση της ιστοσελίδας, τότε η μεν πρώτη (include) θα εκτυπώσει προειδοποιητικό μήνυμα (warning), αλλά θα συνεχίσει να εκτελεί το πρόγραμμα, η δε δεύτερη (require) θα εκτυπώσει μήνυμα λάθους (error) και θα σταματήσει την εκτέλεση.

### 5.2.3 Ένα πρόγραμμα PHP

Τι μονοπάτια είναι αυτά σε ποιο εργαλείο ????

Γράφουμε το παρακάτω κομμάτι κώδικα σε ένα αρχείο:

```
<html><body>
  <?php
      echo ("Αυτό είναι ένα παράδειγμα!");
  ?>
</body></html>
```

Αποθηκεύουμε το αρχείο με κάποιο όνομα (το όνομα του αρχείου δεν πρέπει να έχει κενά) με την κατάληξη .php στον κατάλογο root του εξυπηρετητή μας (στα Windows είναι συνήθως ο κατάλογος "wwwroot" μέσα στον κατάλογο "inetpub" στο δίσκο C: ).

Το επόμενο βήμα είναι να ανοίξουμε το αρχείο στον φυλλομετρητή. Επειδή απαιτείται ο εξυπηρετητής για την εκτέλεση του κώδικα PHP, θα πρέπει να ανοίξουμε το αρχείο μέσω ενός URL που θα εντοπίσει το σωστό αρχείο στον εξυπηρετητή. Στα Windows, θα το βρούμε κοιτώντας τις ρυθμίσεις δικτύου (Network settings) στον πίνακα ελέγχου (Control Panel). Στην καρτέλα "Identification" θα δούμε το "Computer name". Το όνομα του τοπικού υπολογιστή είναι το αρχικό URL. Αν, π.χ. το όνομα του υπολογιστή είναι "mycomp" για να δούμε τα περιεχόμενα του αρχικού καταλόγου γράφουμε "http://mycomp" στο φυλλομετρητή. Για να ανοίξουμε το "test.php" σε έναν κατάλογο που ονομάζεται "examples" μέσα στον αρχικό κατάλογο, θα γράψουμε "`http://mycomp/examples/test.php`" για να δούμε το παράδειγμα.

Αν έχουμε κωδικούς πρόσβασης σε ένα εξυπηρετητή δικτύου που υποστηρίζει PHP, μπορούμε να φορτώσουμε τα αρχεία μας με FTP οπουδήποτε στον εξυπηρετητή.

## Μηνύματα λάθους

Αν δεν λειτουργήσει το παραπάνω παράδειγμα ή κάποιο πρόγραμμα που θα επιχειρήσουμε να τρέξουμε, θα δούμε ένα μήνυμα λάθους όπως αυτό:

```
Parse error: parse error in
C:\Inetpub\wwwroot\documents\test.php on line 12
```

Για κάθε γραμμή κώδικα στην οποία έχει εντοπιστεί κάποιο λάθος θα πάρουμε ένα τέτοιο μήνυμα λάθους. Σε περίπτωση εμφάνισης τέτοιου είδους μηνύματος, ελέγχουμε το αρχείο μας test.php. Τα συχνότερα λάθη είναι η έλλειψη κάποιας ετικέτας τέλους, ενός συμβόλου (;) ή ίσως κάποιων εισαγωγικών.

### 5.2.4 Οι μεταβλητές στην PHP

Οι μεταβλητές στην PHP, όπως και στην Perl, μπορούν να είναι ένας από τους παρακάτω τύπους:

- βαθμωτές (scalar)
- πίνακες (array).

Οι πίνακες μπορεί να είναι: συσχετιζόμενοι (associative array) ή/και πολυδιάστατοι (multidimensional).

Οι μεταβλητές είναι ο κύριος μηχανισμός για τη μεταφορά δεδομένων μεταξύ σελίδων ή τμημάτων σελίδων. Υπάρχουν τρία βασικά πράγματα τα οποία μπορούμε να κάνουμε με μεταβλητές:

- Να τις θέσουμε, δηλαδή να τους δώσουμε μία ή και περισσότερες τιμές.
- Να τις επαναθέσουμε, δηλαδή να τους εκχωρήσουμε άλλη τιμή από αυτή που είχαν ως τώρα.
- Να τις προσπελάσουμε, δηλαδή να διαβάσουμε την τιμή μιας μεταβλητής και μετά να τις επεξεργαστούμε.

**Ονομασία μεταβλητών**

Μπορούμε να ονομάσουμε μία μεταβλητή όπως θέλουμε, αρκεί να λάβουμε υπόψη μας τα ακόλουθα:

- Όλες οι μεταβλητές ξεκινούν με το σύμβολο του δολαρίου $.
- Το όνομά της να αρχίζει με γράμμα
- Να αποτελείται από γράμματα, αριθμούς και τον χαρακτήρα υπογράμμισης ( _ );
- Να μην πρόκειται για κάποιο δεσμευμένο όνομα όπως για παράδειγμα το "print".

***Σημείωση***: Στα ονόματα μεταβλητών έχουν σημασία τα κεφαλαία και τα πεζά, έτσι `$baby_names` και `$Baby_names` δεν είναι τα ίδια.

Στα παραδείγματα που έχουμε δει έως τώρα, οι τιμές των μεταβλητών ήταν κείμενο. Είναι δυνατόν, όμως, να είναι αριθμοί καθώς και άλλα αντικείμενα (objects, arrays, booleans).

Χρησιμοποιούμε απλά ή διπλά εισαγωγικά για να περιβάλουμε κείμενο όπως στο:

```
print ("Αυτό είναι ένα παράδειγμα!");
```

Η παραπάνω εντολή εκτυπώνει την φράση "Αυτό είναι ένα παράδειγμα!". Αν θελήσουμε να εκτυπώσουμε το κείμενο με τα εισαγωγικά πρέπει να χρησιμοποιήσουμε το χαρακτήρα διαφυγής "\", που ορίζει στην PHP να μη χρησιμοποιήσει τον επόμενο χαρακτήρα ως μέρος του κώδικα. Έτσι, για να εκτυπωθεί το κείμενο "Αυτό είναι ένα παράδειγμα!" (με τα εισαγωγικά να φαίνονται στο αποτέλεσμα) θα γράψουμε:

```
print (" \" Αυτό είναι ένα παράδειγμα!\"" );
```

**Βαθμωτές μεταβλητές**

Μία βαθμωτή μεταβλητή μπορεί να περιέχει αριθμούς, γράμματα, φράσεις, κτλ. Οι μεταβλητές αυτές στην PHP ξεκινούν με το σύμβολο ("$"). Η τιμή που έχει κάποια μεταβλητή μπορεί να αλλαχθεί κάθε στιγμή. Στο παρακάτω πρόγραμμα παρουσιάζεται ένα παράδειγμα για την αρχικοποίηση και την επανάθεση μεταβλητών.

```
<?php
  $stuff = "Τι κάνεις;";
   echo ("Γεια σου, $stuff");
   echo ("<p>");
   $stuff = "Πώς σε λένε;";
   echo ("Είμαι μια χαρά, ευχαριστώ! Λοιπόν, $stuff");
?>
```

Όλες οι μεταβλητές ξεκινάνε με "$". Έτσι στον παραπάνω κώδικα αρχικά δημιουργήσαμε μία βαθμωτή μεταβλητή με το όνομα "stuff" και της δώσαμε την τιμή: "Τι κάνεις;", χρησιμοποιώντας τη δήλωση: `$stuff = "Τι κάνεις;";`. Στην συνέχεια εκτυπώνουμε μία φράση σε συνδυασμό με την τιμή που περιέχει η μεταβλητή `$stuff`. Αφού έχουμε δημιουργήσει μία νέα παράγραφο στην ιστοσελίδα μας, με την εκτύπωση της ετικέτας <p>, επαναθέτουμε τη μεταβλητή `$stuff` δίνοντας της την τιμή: "Πώς σε λένε;". Εκτυπώνουμε τη φράση: "Είμαι μια χαρά, ευχαριστώ! Λοιπόν, " σε συνδυασμό με την τιμή της μεταβλητής `$stuff` και τερματίζουμε το πρόγραμμά μας.

Οι κανόνες για τη σύνταξη και τον καθορισμό μεταβλητών είναι:

- να τις θέσουμε τιμή με το σύμβολο = , π.χ. `$stuff = "Τι κάνεις;";`
- να χρησιμοποιούμε εισαγωγικά εάν πρόκειται για χαρακτήρες, π.χ. "Τι κάνεις;". Οι αριθμοί δεν απαιτούν εισαγωγικά.

- να τελειώνουμε κάθε εντολή με το σύμβολο του ερωτηματικού ( ; ).

Καλούμε τη μεταβλητή απλά αναφέροντας το όνομά της.

**Πίνακες**

Οι πίνακες μας δίνουν την δυνατότητα να αποθηκεύσουμε όχι μόνο μία τιμή μέσα σε μία μεταβλητή, αλλά μία σειρά από τιμές.
Αν θελήσουμε να καταγράψουμε όλους τους μαθητές μιας τάξης δημοτικού, θα μπορούσαμε να θέσουμε καθέναν ως μία κανονική μεταβλητή ως εξής:

```
$student1 = "Ανδριαννή";
$student2 = "Ηλλιάνα";
$student3 = "Λευτέρης";
...
```

Ένας πίνακας μας επιτρέπει να αποθηκεύσουμε όλα αυτά σε μία μόνο μεταβλητή, που θα ονομάσουμε $class. Η θέση κάθε στοιχείου μέσα στον πίνακα ονομάζεται "κλειδί". Κάθε στοιχείο του πίνακα έχει το δικό του κλειδί που χρησιμοποιείται για την πρόσβαση του και μπορεί να είναι σειρά γραμμάτων ή αριθμών.

Η συνάρτηση array() εκχωρεί μία σειρά τιμών στον πίνακα μας με τον ακόλουθο τρόπο:

```
$class = array("Ανδριαννή", "Ηλλιάνα", "Λευτέρης");
```

Αυτή η εντολή αποθηκεύει τα ονόματα όλων των μαθητών σε μία μεταβλητή πίνακα την $class. Ο συγκεκριμένος τύπος πίνακα εκχωρεί αυτόματα ένα αριθμημένο κλειδί σε κάθε στοιχείο του πίνακα, δίνοντας στο πρώτο στοιχείο τον αριθμό 0, στο δεύτερο τον αριθμό 1 κ.ο.κ. Έτσι, η Ανδριαννή είναι το στοιχείο [0], η Ηλλιάνα το [1], ο Λευτέρης το [2], κτλ.. Τα κλειδιά είναι ο τρόπος με τον οποίο μπορούν να ανακληθούν οι τιμές των στοιχείων των πινάκων. Μπορεί τώρα να γίνει αναφορά σε οποιοδήποτε στοιχείο του πίνακα ως εξής: $class[2].

Υπάρχει και ένας άλλος τρόπος για να ορίσουμε έναν πίνακα ή ακόμα και να προσθέσουμε στοιχεία σε έναν ήδη υπάρχοντα πίνακα. Ο τρόπος αυτός ορίζει κάθε στοιχείο ξεχωριστά, ως εξής:

```
$class[ ] = "Ανδριαννή";
$class[ ] = "Ηλλιάνα";
$class[ ] = "Λευτέρης";
```

Για να προσθέσουμε ένα νέο μαθητή γράφουμε (ανεξάρτητα από τον τρόπο που χρησιμοποιήσαμε για τη δημιουργία του πίνακα):

```
$class[ ] = "Βασίλης";
```

Η τιμή του κλειδιού του νέου στοιχείου θα εξαρτηθεί από τα ήδη υπάρχοντα κλειδιά. Εάν δεν υπάρχει κάποιο στοιχείο στον πίνακα με κλειδί θετικό ακέραιο αριθμό (συμπεριλαμβανομένου και του 0), τότε το κλειδί του νέου στοιχείου θα είναι το 0. Εάν υπάρχει στοιχείο στον πίνακα με κλειδί θετικό ακέραιο αριθμό (ή το 0), τότε το κλειδί του νέου στοιχείου θα είναι ο μεγαλύτερος ακέραιος που χρησιμοποιείται + 1. Στο συγκεκριμένο παράδειγμα η PHP δίνει αυτόματα στον Βασίλη έναν αριθμό, τον αμέσως επόμενο κενό δηλαδή, που σε αυτή την περίπτωση είναι το [3].

Ο παρακάτω κώδικας θα εκτυπώσει το τρίτο στοιχείο του πίνακα, που είναι ο Λευτέρης. Αξίζει να τονιστεί ότι το πρώτο στοιχείο του πίνακα είναι το `$class[0]`.

```
<?php
print "$class[2]";
?>
```

Εάν χρειάζεται να εκτυπωθούν πληροφορίες για το σύνολο των στοιχείων του πίνακα, τότε χρησιμοποιείται η `print_r`. Για παράδειγμα ο παρακάτω κώδικάς:

```
<?php
print_r ($class);
?>
```

θα έχει ως αποτέλεσμα το:

```
Array
(
    [0] => Ανδριαννή
    [1] => Ηλλιάνα
    [2] => Λευτέρης
    [3] => Βασίλης
)
```

Ορίζοντας και τη δεύτερη παράμετρο εισόδου της print_r μπορούμε να αποθηκεύσουμε το αποτέλεσμα της σε κάποια μεταβλητή. Έτσι στον παρακάτω κώδικας χρησιμοποιείται η μεταβλητή $hight_school ώστε να 'παγιδεύσει' το αποτέλεσμα της εξόδου της print_r.

```
<?php
  $hight_school = print_r ($class, true);
?>
```

**Συσχετιζόμενοι πίνακες**

Οι συσχετιζόμενοι πίνακες διαχωρίζουν τα περιεχόμενα στοιχεία όχι με αριθμούς, αλλά με λεκτικά που καθορίζει ο προγραμματιστής. Το ζευγάρι κλειδί και τιμή (key -value), ορίζουν πλήρως κάθε στοιχείο ενός πίνακα. Χρησιμοποιώντας τη συνάρτηση array() καθορίζονται ζεύγη με το όνομα του κλειδιού και την τιμή του στοιχείου. Η σύνδεση κλειδιού και τιμής πραγματοποιείται με το σύμβολο "=>", π.χ. key=>"value". Για παράδειγμα ο κώδικας:

```
$students = array (
            "name"=>"Τονυ",
            "haircolor"=>"μαύρα",
            "eyecolor"=>"καστανά",
            "age"=>5);
```

ορίζει στον πίνακα τη δημιουργία των κλειδιών "name", "haircolor" "eyecolor" και "age" και σε κάθε κλειδί σχετίζει μία συγκεκριμένη τιμή ("Τονυ","μαύρα","καστανά",5).

Η αναφορά σε οποιοδήποτε στοιχείο του πίνακα, πραγματοποιείται μέσω των ονομάτων των κλειδιών. Για παράδειγμα το παρακάτω:

```
print $students[eyecolor];
```

θα εκτυπώσει στην οθόνη του χρήστη την τιμή "καστανά". Η τιμή του κάθε κλειδιού μπορεί να αλλαχθεί και μεμονωμένα, όπως παρουσιάζεται παρακάτω:

```
$students["name"] = "Ντένη";
$students["haircolor"] = "μαύρα";
$students["eyecolo"r] = "καστανά";
$students["age"] = 6;
```

**Πολυδιάστατοι πίνακες**

Ένας πολυδιάστατος πίνακας είναι ένας πίνακας (π.χ. οι μαθητές της τάξης του δημοτικού) φτιαγμένος από άλλους πίνακες (π.χ. για κάθε μαθητή, ένας πίνακας που περιέχει το όνομα, το χρώμα των μαλλιών, το χρώμα των ματιών και την ηλικία του μαθητή). Δημιουργούμε πολυδιάστατους πίνακες όπως παρακάτω:

```
$students = array (                );
```

Γεμίζουμε τον πίνακα αυτό με άλλους, στους οποίους έχουν προσδιοριστεί τα κλειδιά, ως εξής:

```
$students = array (
           array ("name"=>"Τόνυ",
                     "haircolor"=>"μαύρα",
                     "eyecolor"=>"καστανά",
                     "age"=>5),
           array ("name"=>"Ντένη",
                     "haircolor"=>"καστανά",
                     "eyecolor"=>"καφέ",
                     "age"=>6 ),
           array ("name"=>"Δροσούλα",
                     "haircolor"=>"μαύρα",
                     "eyecolor"=>"καστανά σκούρα",
                     "age"=>11 ),
           array ("name"=>"Βασίλης",
                     "haircolor"=>"ξανθά",
                     "eyecolor"=>"πράσινα",
                     "age"=>7 )
                              );
```

Στον πίνακα αυτό, μπορούμε να πάρουμε οποιοδήποτε μέρος της πληροφορίας που περιλαμβάνεται, δίνοντας το όνομα του πρώτου πίνακα `$students` (του πίνακα που περιέχει τους άλλους), τη θέση του υπο-πίνακα και μετά το όνομα του κλειδιού για το χαρακτηριστικό που θέλουμε να πάμε.
Για να βρούμε την ηλικία της "`Δροσούλας`" θα γράψουμε:

```
print $students [2]["age"];
```

### 5.2.5 Σταθερές

Παρόμοιος μηχανισμός με τις μεταβλητές για την αποθήκευση / κατοχή πληροφοριών, αποτελούν και οι σταθερές (constants). Υπάρχει, όμως, μια ουσιαστική διαφορά, σε σχέση με τις μεταβλητές. Στις σταθερές δεν είναι δυνατή η αλλαγή της τιμής τους, από τη στιγμή που τους έχει οριστεί, δηλαδή τους έχει

εκχωρηθεί, κάποια αρχική τιμή. Για παράδειγμα ο παρακάτω κώδικας ορίζει τη σταθερά "ROOT_LOCATION".

```
define("ROOT_LOCATION", "/usr/local/www/");
```

Γενικά προτιμάται τα ονόματα των σταθερών να είναι με κεφαλαία γράμματα ώστε να ξεχωρίζει, έτσι, ο ρόλος τους.

***Σημείωση**: Εξ ορισμού, μια σταθερά μπορεί να χρησιμοποιηθεί οπουδήποτε μέσα στο αρχείο. Έχουν, δηλαδή, 'καθολική' εμβέλεια εφαρμογής (global scope).*

### 5.2.6 Τελεστές

Οι τελεστές χρησιμοποιούνται για πράξεις ανάμεσα στις μεταβλητές. Υπάρχουν διαφορετικοί τύποι τελεστών, όπως: εκχώρησης, σύγκρισης, λογικοί και αριθμητικοί.

**Τελεστές εκχώρησης**

Είδαμε σε προηγούμενη παράγραφο τον τρόπο που χρησιμοποιείται το σύμβολο "=" στον ορισμό των μεταβλητών στα σενάρια/προγράμματα της PHP. Το σύμβολο αυτό ονομάζεται τελεστής εκχώρησης και είναι ο πιο απλός τελεστής. Για παράδειγμα το: `$a = b`, σημαίνει ότι «στη μεταβλητή a εκχωρείται η τιμή b».

Πολυπλοκότεροι τελεστές εκχώρησης περιλαμβάνουν την πρόσθεση της ποσότητας που βρίσκεται στην δεξιά πλευρά του τελεστή, στη μεταβλητή που υπάρχει αριστερά. Πρόκειται για τον τελεστή '+=' . Για παράδειγμα οι παρακάτω εντολές έχουν το ίδιο αριθμητικό αποτέλεσμα.

```
$a +=4;
$a=$a+4;
```

Στον Πίνακας 5.1.1 παρουσιάζονται οι τελεστές εκχώρησης της PHP.

**Πίνακας 5.1 Τελεστές εκχώρησης στην PHP**

| Τελεστής | Περιγραφή |
|---|---|

| | |
|---|---|
| = | Εκχωρεί την τιμή δεξιά του συμβόλου στη μεταβλητή που υπάρχει αριστερά. |
| += | Προσθέτει τη ποσότητα που βρίσκεται στην δεξιά πλευρά του τελεστή, στη μεταβλητή που υπάρχει αριστερά. |
| -= | Αφαιρεί την ποσότητα που βρίσκεται στην δεξιά πλευρά του τελεστή, στη μεταβλητή που υπάρχει αριστερά. |
| *= | Πολλαπλασιάζει την ποσότητα που βρίσκεται στην δεξιά πλευρά του τελεστή με τη μεταβλητή που υπάρχει αριστερά. Το αποτέλεσμα εκχωρείται στη μεταβλητή που υπάρχει αριστερά. |
| /= | Διαιρεί τη μεταβλητή που βρίσκεται στην αριστερά πλευρά του τελεστή με τη ποσότητα που υπάρχει δεξιά. Το αποτέλεσμα εκχωρείται στη μεταβλητή που υπάρχει αριστερά. |
| .= | Προσθέτει / συνδυάζει την αλφαριθμητική τιμή (string) που βρίσκεται στην δεξιά πλευρά του τελεστή με τη μεταβλητή που υπάρχει αριστερά. Το αποτέλεσμα εκχωρείται στη μεταβλητή που υπάρχει αριστερά. Χρησιμοποιείται για αλφαριθμητικές (string) τιμές και μεταβλητές. |
| %= | Αποδίδει το υπόλοιπο (modulo) μετά τη διαίρεση της μεταβλητή που βρίσκεται στην αριστερά πλευρά του τελεστή με τη ποσότητα που υπάρχει δεξιά. Το αποτέλεσμα εκχωρείται στη μεταβλητή που υπάρχει αριστερά. |

**Τελεστές σύγκρισης**

Οι τελεστές σύγκρισης δίνουν τη δυνατότητα να συγκρίνουμε αν τα στοιχεία είναι ίσα, πανομοιότυπα, μικρότερο ή μεγαλύτερο το ένα από το άλλο. Συνήθως χρησιμοποιούνται σε συνδυασμό με τις εντολές υπό συνθήκη, π.χ. if. Περισσότερα για τις εντολές αυτές θα γνωρίσουμε αργότερα. Στον Πίνακα 5.3.1 παρουσιάζονται οι τελεστές σύγκρισης όπως αυτοί χρησιμοποιούνται στην PHP.

**Πίνακας 5.2 Τελεστές σύγκρισης στην PHP**

| Τελεστής | Περιγραφή |
|---|---|
| == | Επιστρέφει την τιμή ‘αληθής’ (true), εάν η τιμή της μεταβλητής στα αριστερά είναι ίση με αυτή στα δεξιά. Αλλιώς επιστρέφει την τιμή ‘ψευδής’ (false). |

| | |
|---|---|
| === | Επιστρέφει την τιμή ‘αληθής’ (true), εάν η τιμή της μεταβλητής αριστερά είναι ίση με αυτή στα δεξιά, και είναι και του ίδιου τύπου. Αλλιώς επιστρέφει την τιμή ‘ψευδής’ (false). |
| != | Επιστρέφει την τιμή ‘αληθής’ (true), εάν η τιμή της μεταβλητής αριστερά είναι δεν ίση με αυτή στα δεξιά. Αλλιώς επιστρέφει την τιμή ‘ψευδής’ (false). |
| <> | Επιστρέφει την τιμή ‘αληθής’ (true), εάν η τιμή της μεταβλητής αριστερά είναι δεν ίση με αυτή στα δεξιά. Αλλιώς επιστρέφει την τιμή ‘ψευδής’ (false). |
| !== | Επιστρέφει την τιμή ‘αληθής’ (true), εάν η τιμή της μεταβλητής αριστερά είναι δεν ίση με αυτή στα δεξιά και δεν είναι και του ίδιου τύπου. Αλλιώς επιστρέφει την τιμή ‘ψευδής’ (false). |
| < | Επιστρέφει την τιμή ‘αληθής’ (true), εάν η τιμή της μεταβλητής αριστερά είναι μικρότερη από αυτή στα δεξιά. Αλλιώς επιστρέφει την τιμή ‘ψευδής’ (false). |
| > | Επιστρέφει την τιμή ‘αληθής’ (true), εάν η τιμή της μεταβλητής αριστερά είναι μεγαλύτερη από αυτή στα δεξιά. Αλλιώς επιστρέφει την τιμή ‘ψευδής’ (false). |
| <= | Επιστρέφει την τιμή ‘αληθής’ (true), εάν η τιμή της μεταβλητής αριστερά είναι μικρότερη ή ίση με αυτή στα δεξιά. Αλλιώς επιστρέφει την τιμή ‘ψευδής’ (false). |
| >= | Επιστρέφει την τιμή ‘αληθής’ (true), εάν η τιμή της μεταβλητής αριστερά είναι μεγαλύτερη ή ίση με αυτή στα δεξιά. Αλλιώς επιστρέφει την τιμή ‘ψευδής’ (false). |

***Σημείωση**: Συχνά προκαλείται σύγχυση ανάμεσα στον τελεστή εκχώρησης = και στους τελεστές σύγκρισης τιμής ==, αλλά και τύπου ===.*

***Συμβουλή**: Επειδή είναι δυνατόν κατά τη χρήση των τελεστών σύγκρισης να πραγματοποιηθεί εσφαλμένα εκχώρηση τιμής, προτείνεται η τιμή να τοποθετείται στα αριστερά του τελεστή σύγκρισης. Για παράδειγμα αντί του if ($var == 12) { ... } να χρησιμοποιείται το if (12 == $var) { ... }*

## Λογικοί Τελεστές

Οι λογικοί τελεστές, μπορεί να θεωρηθεί, ότι χρησιμοποιούνται ώστε να συνδυαστούν τα αποτελέσματα δυο τελετών σύγκρισης. Για παράδειγμα ο λογικός τελεστής and μπορεί να δώσει απάντηση στην ερώτηση «είναι ($a>1 and

$a<2) τότε κάνε κάτι». Όπως φαίνεται προσθέτει λογικά τις δυο συγκρίσεις ($a>1, $a<2) σε μια ερώτηση. Ως γενικός κανόνας προκύπτει ότι κάθε φράση/μεταβλητή/συνθήκη που μπορεί να πάρει τιμή "αληθής" ή «ψευδής», τότε μπορεί να χρησιμοποιηθεί με τους λογικούς τελεστές. Κάθε τελεστής τέτοιου είδους μπορεί να δεχθεί δυο παραμέτρους εισόδου (συγκρίσεις), ο οποίες έχουν αποτέλεσμα "αληθές" ή «ψευδές» και να παράγει ένα.

Στον πίνακα 5.3.2 παρουσιάζονται οι λογικοί τελεστές όπως αυτοί χρησιμοποιούνται στην PHP.

**Πίνακας 5.3 Λογικοί τελεστές στην PHP**

| Τελεστής | Περιγραφή |
|---|---|
| and | Επιστρέφει την τιμή 'αληθής' (true), εάν αν η τιμή της μεταβλητής αριστερά και στα δεξιά είναι 'αληθής'. Αλλιώς αν κάποια από τις δύο είναι 'ψευδής' επιστρέφει την τιμή 'ψευδής' (false). |
| or | Επιστρέφει την τιμή 'αληθής' (true), εάν η τιμή μιας μεταβλητής είναι 'αληθής' είτε και οι δύο είναι 'αληθής'. Αλλιώς αν και οι δύο είναι 'ψευδής' επιστρέφει την τιμή 'ψευδής' (false). |
| xor | Επιστρέφει την τιμή 'αληθής' (true), εάν η τιμή μιας μεταβλητής (αριστερά ή στα δεξιά) είναι 'αληθής'. Αλλιώς αν και οι δύο είναι 'ψευδής' ή 'αληθής', επιστρέφει την τιμή 'ψευδής' (false). |
| ! | Επιστρέφει την τιμή 'αληθής' (true), εάν η τιμή της μεταβλητής δεν είναι 'ψευδής'. Αλλιώς αν η τιμή της μεταβλητής είναι 'αληθής' επιστρέφει την τιμή 'ψευδής' (false). |
| && | Επιστρέφει την τιμή 'αληθής' (true), εάν η τιμή της μεταβλητής αριστερά και στα δεξιά είναι 'αληθής'. Αλλιώς αν κάποια από τις δύο είναι 'ψευδής' επιστρέφει την τιμή 'ψευδής' (false). Ίδιος με τον τελεστή 'and'. |
| \|\| | Επιστρέφει την τιμή 'αληθής' (true), εάν η τιμή της μεταβλητής αριστερά ή στα δεξιά είναι 'αληθής', είτε είναι 'αληθής' και οι δύο. Αλλιώς αν και οι δύο είναι 'ψευδής' επιστρέφει την τιμή 'ψευδής' (false). Ίδιος με τον τελεστή 'or'. |

## Αριθμητικοί Τελεστές

Οι αριθμητικοί τελεστές, όπως φανερώνει και το όνομά τους, χρησιμοποιούνται για την πραγματοποίηση αριθμητικών πράξεων. Οι πράξεις αυτές μπορεί να είναι απλές μαθηματικές, όπως πρόσθεση, αφαίρεση,

πολλαπλασιασμός και διαίρεση. Ενδέχεται να είναι πολυπλοκότερες όπως το υπόλοιπο διαίρεσης (modulo) και η αύξηση ή αφαίρεση της τιμής μιας μεταβλητής.

Στον πίνακα 5.3.3 παρουσιάζονται οι αριθμητικοί τελεστές όπως αυτοί χρησιμοποιούνται από την PHP.

**Πίνακας 5.4 Αριθμητικοί τελεστές στην PHP**

| Τελεστής | Περιγραφή |
|---|---|
| + | Αθροίζει δύο τιμές. |
| - | Αφαιρεί δύο τιμές. |
| * | Πολλαπλασιάζει δύο τιμές. |
| / | Διαιρεί δύο τιμές. |
| % | Παρουσιάζει το υπόλοιπο της διαίρεσης δύο αριθμών. |
| ++ | Αυξάνει την τιμή κατά ένα. |
| -- | Ελαττώνει την τιμή κατά ένα. |

***Σημείωση**: Ανάλογα με τη θέση της μεταβλητής σε σχέση με τον αριθμητικό τελεστή ++, επιστρέφεται και διαφορετικό αποτέλεσμα. Για παράδειγμα αν ισχύει το ++$a, τότε αυξάνεται η τιμή κατά μια μονάδα και στη συνέχεια επιστρέφεται/αποδίδεται η μεταβλητή προς χρήση. Ενώ αν ισχύει το $a++, τότε, πρώτα επιστρέφεται και στη συνέχεια αυξάνεται η τιμή της μεταβλητής κατά μια μονάδα Το ανωτέρω χρησιμοποιείται συχνά στους βρόχους επανάληψης.*

### 5.2.7 Εντολές πολλαπλών γραμμών

Υπάρχουν περιπτώσεις όπου χρειάζεται να εκτυπωθεί κείμενο, το οποίο καταλαμβάνει πολλαπλές γραμμές. Η χρήση πολλών επαναλαμβανόμενων συναρτήσεων print και echo, σε αυτή την περίπτωση, μπορεί να αποβεί χρονοβόρα καθώς και επιρρεπής σε λάθη. Επιπλέον, ένα μεγάλο κείμενο μπορεί να χρειάζεται να εκχωρηθεί σε κάποια μεταβλητή (string). Για να ξεπεραστούν τέτοιου είδους δυσκολίες η PHP προσφέρει δυο λύσεις..

Πρώτα μπορούν να χρησιμοποιηθούν τα διπλά εισαγωγικά, τα οποία θα διατηρήσουν τη μορφοποίηση. Έτσι, εάν, περιβάλουμε το κείμενο με διπλά εισαγωγικά τότε αυτό μπορεί να επεκταθεί σε πολλές γραμμές.

Για παράδειγμα ο παρακάτω κώδικας θα εκτυπώσει το κείμενο όπως παρουσιάζεται στην Εικόνα 5.1.

```
<?php
$word_of_the_day="πολλαπλός -ή -ό";

echo "Η λέξη της ημέρας: $word_of_the_day

$word_of_the_day : που συντίθεται από πολλά στοιχεία ή μέρη,
που παρουσιάζεται με πολλές μορφές,
που γίνεται με πολλούς τρόπους,
που επιτελεί πολλές λειτουργίες,
σύνθετος.";
?>
```

***Σημείωση**: Οι μεταβλητές μεταφράζονται στις τιμές τους όταν περιλαμβάνονται σε διπλά εισαγωγικά, ενώ δεν ισχύει το ίδιο όταν χρησιμοποιούνται μονά εισαγωγικά.*

```
Η λέξη της ημέρας: πολλαπλός -ή -ό

πολλαπλός -ή -ό : που συντίθεται από πολλά στοιχεία ή μέρη,
που παρουσιάζεται με πολλές μορφές,
που γίνεται με πολλούς τρόπους,
που επιτελεί πολλές λειτουργίες,
σύνθετος.
```

**Εικόνα 5.1 – Εκτύπωση πολλαπλών γραμμών**

Ένας διαφορετικός τρόπος περιλαμβάνει τη χρήση του τελεστή '<<<', ο οποίος ορίζει ότι μια αλληλουχία χαρακτήρων (λεκτικό) θα περικλείει ένα κείμενο, διατηρώντας τις εναλλαγές γραμμής, και τυχόν διαστήματα (περιλαμβανομένου και των εσοχών). Εφαρμόζοντας τον τελεστή το ανωτέρω παράδειγμα γίνεται:

```
<?php
```

```
$word_of_the_day="πολλαπλός -ή -ό";


echo <<<_PAPEI
Η λέξη της ημέρας: $word_of_the_day


$word_of_the_day : που συντίθεται από πολλά στοιχεία ή μέρη,
που παρουσιάζεται με πολλές μορφές,
που γίνεται με πολλούς τρόπους,
που επιτελεί πολλές λειτουργίες,
σύνθετος.
_PAPEI;


?>
```

Το αποτέλεσμα στην οθόνη του φυλλομετρητή είναι το ίδιο (Εικόνα 5.1).

***Σημείωση**: Ο τελεστής <<< ονομάζεται ΄ευθύγραμμη εισαγωγή΄ (here document ή εν συντομία heredoc).*

Ο,τιδήποτε περιλαμβάνεται ανάμεσα στα λεκτικά _PAPEI θα αποδοθεί στην οθόνη του χρήστη σα να περικλειόταν σε διπλά εισαγωγικά. Η χρήση αυτού του τελεστή καθιστά δυνατή την εκτύπωση ολόκληρης ιστοσελίδας HTML.
Για την ορθή λειτουργία του τελεστή επιβάλλεται η έναρξη (echo <<<_PAPEI) και η λήξη (_PAPEI;) του λεκτικού να αποτελούν το μοναδικό στοιχείο στη γραμμή. Δεν επιτρέπεται η ύπαρξη ούτε ενός κενού διαστήματος μετά το λεκτικό. Η επιλογή του λεκτικού αποτελεί απόφαση του προγραμματιστή. Εδώ χρησιμοποιήθηκε το λεκτικό '_PAPEI'. Θα μπορούσε να είναι οτιδήποτε ακόμα και το όνομα του προγραμματιστή. Γενικά καλό είναι τα λεκτικά αυτά να ξεκινάνε με κάτω παύλα (_), ώστε να αποφεύγεται, τυχόν, σύγχυση με ονόματα συναρτήσεων ή μεταβλητών.

### 5.2.8 Εντολές διακλάδωσης στην PHP

Όλη η ιδέα των δυναμικών ιστοσελίδων βρίσκεται στο να γίνει η ιστοσελίδα όσο πιο 'έξυπνη' γίνεται. Η εφαρμογή θα πρέπει να μπορεί να "πάρει"

αποφάσεις βασισμένη στα διαφορετικά δεδομένα εισόδου που παρέχει ο κάθε χρήστης. Παραδείγματα τέτοιων διαφορετικών αποφάσεων είναι τα παρακάτω:

- Αφού ένας χρήστης εισάγει μία διεύθυνση ηλεκτρονικού ταχυδρομείου, να ελέγχει εάν αυτή έχει τη σωστή μορφή (whoever@wherever.com) και αν δεν έχει τη μορφή αυτή, να δείχνει μία σελίδα που να λέει π.χ. "Please, enter a valid email address!"
- Να εξυπηρετεί δύο ή και περισσότερες διευθύνσεις, π.χ. μία με κατάληξη .com και μία άλλη με κατάληξη .gr.
- Να αναγνωρίζει πότε ένας πελάτης σε ηλεκτρονικό κατάστημα χρησιμοποιεί τραπεζικό λογαριασμό για μία ηλεκτρονική πληρωμή ή πιστωτική κάρτα και να εμφανίζει την ανάλογη φόρμα πληρωμής.

Μια δυναμική ιστοσελίδα θα πρέπει να αναγνωρίζει διαφορετικές περιπτώσεις/καταστάσεις και να παράγει ανάλογα αποτελέσματα. Αυτό πραγματοποιείται με τη χρήση δηλώσεων / εντολών διακλάδωσης σε συνδυασμό με τελεστές, κυρίως: σύγκρισης και λογικούς. Ως εντολές διακλάδωσης αναφέρονται οι: `if`, `else`, και `elseif`, `switch` και ο τελεστής `?`.

**Εντολή if**

Η δήλωση `if` δίνει τη δυνατότητα να αποδώσουμε με κώδικα τις παρακάτω καταστάσεις :

```
ΑΝ κάποια συνθήκη είναι αληθής, τότε κάνε κάτι.
ΑΝ η συνθήκη είναι ψευδής, τότε αγνόησέ το.
```

Η σύνταξη της εντολής `if` είναι η ακόλουθη:

```
if (συνθήκη) {
/* Ο κώδικας αυτός θα εκτελεστεί εάν η συνθήκη είναι αληθής
*/
}
```

Ως συνθήκη μπορεί να είναι κάθε έγκυρη έκφραση της PHP, όπως κάποια ισότητα ή σύγκριση, έλεγχος αν κάποιο στοιχείο είναι 0 ή NULL, κάποια συνάρτηση.

Εάν οι εντολές που πρόκειται να εκτελεστούν εφόσον ισχύει η συνθήκη της `if`, είναι περισσότερες από μία, τότε, αυτές περικλείονται σε αγκύλες. Διαφορετικά εάν πρόκειται για μια μόνο εντολή, οι αγκύλες μπορεί και να παραληφθούν. Αποτελεί καλή πρακτική να χρησιμοποιούνται πάντοτε οι

αγκύλες. Επιπλέον, μερικοί προγραμματιστές, τοποθετούν την εναρκτήρια αγκύλη στην ίδια γραμμή με την εντολή if, ενώ άλλοι στην επόμενη γραμμή. Οποιοσδήποτε από αυτούς του δύο τρόπους είναι σωστός.
Στο παρακάτω παράδειγμα πρώτα θέτουμε τη μεταβλητή $FavoriteDay ίση με την τιμή «Παρασκευή». Στη συνέχεια εξετάζουμε τη συνθήκη, εάν η τιμή της μεταβλητής $FavoriteDay είναι ίση με «Παρασκευή» και εφόσον είναι ίση εκτυπώνεται η φράση «Μου αρέσει η Παρασκευή πολύ!!».

```
<?php
$FavoriteDay = "Παρασκευή";
if ($FavoriteDay == "Παρασκευή") {
print ("Μου αρέσει η Παρασκευή πολύ!!");
}
?>
```

## Εντολή else

Υπάρχουν περιπτώσεις όπου χρειάζεται να ελεγχθεί η ορθότητα μιας συνθήκης αλλά ταυτόχρονα και η σφαλερότητα της. Και στις δυο περιπτώσεις (ισχύει / δεν ισχύει, σωστό / λάθος) θα πρέπει να οριστεί στο πρόγραμμα η εκτέλεση κάποιων εντολών. Το ανωτέρω πραγματοποιείται με τον συνδυασμό των εντολών if και else. Η εντολή else ενσωματώνεται στη δήλωση του if και παρέχεται, έτσι, η δυνατότητα απόδοσης της παρακάτω κατάστασης, ως εξής:

```
ΑΝ κάποια συνθήκη είναι αληθής, κάνε κάτι
ΑΛΛΙΩΣ, αν η συνθήκη δεν είναι ψευδής,
κάνε κάτι άλλο.
```

Η σύνταξη είναι η ακόλουθη:

```
if (συνθήκη) {
/* Ο κώδικας αυτός θα εκτελεστεί εάν η συνθήκη είναι αληθής
*/
} else {
/* Ο κώδικας αυτός θα εκτελεστεί εάν η συνθήκη είναι ψευδής
*/
}
```

Στο συνδυασμό των εντολών if και else, εάν η συνθήκη είναι ‘αληθής’ τότε εκτελούνται οι εντολές της if . Διαφορετικά, εάν η συνθήκη είναι ‘ψευδής’, τότε εκτελούνται οι εντολές της else. Σε κάθε περίπτωση θα ισχύει ή το ένα (αληθής)

ή το άλλο (ψευδής) ή τίποτα από τα δύο (άκυρη συνθήκη). Σε καμία περίπτωση δεν μπορούν να συμβούν και τα δυο ταυτόχρονα, δηλαδή να είναι η συνθήκη και αληθής και ψευδής.
Ένα παράδειγμα που χρησιμοποιεί το `else` είναι το παρακάτω, όπου ελέγχεται η τιμή της μεταβλητής `$FavoriteDay` και αν ισούται με «`Σάββατο`», τότε εκτυπώνεται το μήνυμα `"Μου αρέσει το Σάββατο πολύ!!"`. Διαφορετικά, εάν η τιμή της μεταβλητής `$FavoriteDay` και δεν ισούται με «`Σάββατο`» εκτυπώνεται το μήνυμα `"Δεν μου αρέσει το Σάββατο."`.

```
<?php
$FavoriteDay = "Παρασκευή";
if ($FavoriteDay == "Σάββατο") {
print ("Μου αρέσει το Σάββατο πολύ!!");
} else {
print ("Δεν μου αρέσει το Σάββατο.");
}
?>
```

**Εντολή elseif**

Μερικές φορές υπάρχουν περιπτώσεις όπου χρειάζεται να ελεγχθούν περισσότερες από μια συνθήκες. Αυτό πραγματοποιείται με τη χρήση της εντολής `elseif` σε συνδυασμό με τις `if` και `else`. Όπως φανερώνει και το όνομά της, είναι παρόμοια με την `else`, με τη διαφορά ότι προσθέτει μια επιπλέον συνθήκη. Η `elseif` μπορεί να περιγραφεί ως:

```
ΑΝ κάποια συνθήκη είναι αληθής,
κάνε κάτι.
ΑΛΛΙΩΣ ΑΝ κάποια διαφορετική συνθήκη είναι αληθής,
κάνε κάτι.
ΑΛΛΙΩΣ, αν όλες οι προηγούμενες συνθήκες είναι ψευδής,
κάνε κάτι άλλο.
```

Μέσα στο συνδυασμό `if` και `else`, μπορούν να υπάρξουν όσα `elseif` χρειάζονται. Δηλαδή μπορεί να ισχύει `if ….elseif ……elseif ……elseif ……else`. Γενικά δεν αποτελεί καλή πρακτική η χρήση πολλαπλών `elseif`.
Θα εκτελεστούν οι εντολές τις πρώτης συνθήκης που θα βρεθεί να είναι ‘αληθής’.

***Σημείωση**: Εφόσον οι εντολές που είναι προς εκτέλεση περικλείονται σε αγκύλες, τότε είναι δυνατόν η εντολή* `elseif` *να γραφεί με δύο λέξεις, δηλαδή* `else if`.

Στο παρακάτω παράδειγμα φαίνεται η χρήση της elseif. Όπως και στα προηγούμενα παραδείγματα, ελέγχεται η τιμή της μεταβλητής $FavoriteDay και αν ισούται με "Σάββατο", τότε εκτυπώνεται το μήνυμα "Μου αρέσει το Σάββατο πολύ!!". Διαφορετικά, (εάν η τιμή της μεταβλητής $FavoriteDay και δεν ισούται με "Σάββατο") ελέγχεται εάν η μεταβλητή ισούται με "Κυριακή", οπότε και εκτυπώνει το κατάλληλο μήνυμα.

```
<?php
$FavoriteDay = "Κυριακή";

if ($FavoriteDay== "Σάββατο") {
print ("Μου αρέσει το Σάββατο πολύ!!");
} elseif ($FavoriteDay == "Κυριακή") {
print ("Επιτέλους, Κυριακή!");
}
?>
```

Το τελικό else μπορεί και να παραληφθεί, όπως φαίνεται στο παράδειγμά μας. Θα μπορούσαμε, όμως, να προστεθεί μια εντολή else στο τέλος για να καλυφθεί η περίπτωση όπου η μεταβλητή $FavoriteDay δεν έχει την τιμή "Σάββατο", ούτε την τιμή "Κυριακή".

## Εντολή switch

Η εντολή switch είναι χρήσιμη σε περιπτώσεις κατά τις οποίες μια μεταβλητή ή το αποτέλεσμα μιας έκφρασης, ενδέχεται να 'πάρει' διαφορετικές τιμές. Ανάλογα με την τιμή που ισχύει θα πρέπει να εκτελεστούν και διαφορετικές εντολές.

Η switch μπορεί να περιγραφεί ως:

```
Έλεγξε αν η συνθήκη είναι:
      Ίση με κάποια τιμή
            Τότε κάνε κάτι
      Ίση με κάποια δεύτερη τιμή
            Τότε κάνε κάτι
      Ίση με κάποια τρίτη τιμή
            Τότε κάνε κάτι
      Ίση με κάποια τέταρτη τιμή
            Τότε κάνε κάτι
            ……..
      Αν δεν ισχύει τίποτα από τα παραπάνω
            Κάνε κάτι άλλο
```

Ο παρακάτω κώδικας αποτελεί παράδειγμα της χρήσης της `switch`, όπου ελέγχεται η τιμή της μεταβλητής `$FavoriteDay` σε σχέση με τα ονόματα των ημερών της εβδομάδας. Η συνθήκη, σε αυτή την περίπτωση, είναι η μεταβλητή `$FavoriteDay`, η οποία ελέγχεται (`case`) αν είναι ίση με το όνομα κάποιας ημέρας. Αν βρεθεί ότι υπάρχει ισότητα, στο παράδειγμα με τη λέξη `"Σάββατο"`, τότε εκτυπώνει ένα μήνυμα και διακόπτεται (`break`) η συνέχιση της εντολής `switch`.

```
<?php
$FavoriteDay = "Σάββατο";

switch ($FavoriteDay) {
    case "Δευτέρα":
        echo "Μου αρέσει η Δευτέρα πολύ!!";
        break;
    case "Τρίτη":
        echo "Μου αρέσει η Τρίτη πολύ!!";
        break;
    case "Τετάρτη":
        echo "Μου αρέσει η Τετάρτη πολύ!!";
        break;
    case "Πέμπτη":
        echo "Μου αρέσει η Πέμπτη πολύ!!";
        break;
    case "Παρασκευή":
        echo "Μου αρέσει η Παρασκευή πολύ!!";
        break;
    case "Σάββατο":
        echo "Μου αρέσει το Σάββατο πολύ!!";
        break;
    case "Κυριακή":
```

```
        echo "Μου αρέσει η Κυριακή πολύ!!";
        break;
    default:
        echo "Δεν έχω κάποια προτίμηση.";
}
?>
```

Οι κανόνες σύνταξης της `switch`, δεν υποχρεώνουν στη χρήση αγκύλων μέσα στο τμήμα της `case`. Οι εντολές που είναι προς εκτέλεση περιλαμβάνονται στο σύμβολο `":"` και την εντολή `break`. Αντίθετα αγκύλες χρησιμοποιούνται κατά την έναρξη της `switch` και τον τερματισμό της.

Η χρήση της `break` είναι σημαντική, καθώς αποτρέπει την ανεξέλεγκτη εκτέλεση της `switch`. Στο ανωτέρω παράδειγμα, αν δεν υπήρχε η εντολή `break`, τότε το πρόγραμμα θα εκτύπωνε το μήνυμα `"Μου αρέσει το Σάββατο πολύ!!"` (αφού η μεταβλητή είναι ίση με `"Σάββατο"`) και στη συνέχεια θα εκτελούσε και τον επόμενο στη σειρά έλεγχο (στην περίπτωσή μας τη λέξη `"Κυριακή"`) και αν υπήρχε ταύτιση θα εκτελούσε και αυτές τις εντολές. Η χρήση της `break` τερματίζει την εκτέλεση της `switch`.
Είναι δυνατόν η συνθήκη που ελέγχεται μέσω της `switch` να μην έχει ισότητα με κάποια από τις προτεινόμενες τιμές. Σε αυτήν την περίπτωση θα εκτελεστούν οι εντολές που περιλαμβάνονται στο τμήμα `"default"`. Εάν η `"default"` τοποθετηθεί στο τέλος της `switch` δε χρειάζεται να περιλαμβάνει και κάποια εντολή `break`. Σε διαφορετική περίπτωση, εάν δηλαδή τοποθετηθεί ανάμεσα στις `case`, τότε επιβάλλεται η χρήση της `break`.

**Εντολή διακλάδωσης με ?**

Ο τελεστής `?` χρησιμοποιείται ώστε αν αξιολογηθεί μια έκφραση ως αληθής ή ψευδής και να εκτελεστούν οι αντίστοιχες εντολές. Αποτελεί, δηλαδή, μια σύντομη εκδοχή της εντολής

```
if (έκφραση ==TRUE) { εκτέλεσε_εντολές για TRUE;  }
else {εκτέλεσε_εντολές για FALSE; }
```

Χρησιμοποιείται ώστε να αποδοθεί με κώδικα η κατάσταση:

```
έκφραση ? εντολές για TRUE: εντολές για FALSE;
```

Το παρακάτω παράδειγμα κώδικα παρουσιάζει τη χρήση του τελεστή ?. Συγκεκριμένα χρησιμοποιείται σε συνδυασμό με την εντολή εκτύπωσης echo. Αν είναι αληθής η συνθήκη: $FavoriteDay=="Σάββατο", τότε θα επιστραφεί, (άρα και θα εκτυπωθεί μέσω της echo) το μήνυμα "Μου αρέσει το Σάββατο πολύ!!". Στην περίπτωση όπου η συνθήκη: $FavoriteDay=="Σάββατο" αξιολογηθεί ως ψευδής, θα επιστραφεί το μήνυμα "Δεν έχω κάποια προτίμηση.".

```
<?php
echo $FavoriteDay=="Σάββατο" ? "Μου αρέσει το Σάββατο
πολύ!!" : "Δεν έχω κάποια προτίμηση.";
?>
```

Ο τρόπος αυτός είναι εξαιρετικά χρήσιμος ώστε να εκφραστούν συγκρίσεις που καταλαμβάνουν μια γραμμή (decision inline). Η χρήση του έχει ως αποτέλεσμα έναν συνοπτικό και σύντομο κώδικα.

***Σημείωση**: Ο τελεστής ? καλείται και τριαδικός (ternary), καθώς δέχεται τρία ορίσματα, σε σχέση με τα συνήθη δύο.*

### 5.2.9 Βρόχοι επανάληψης

Οι βρόχοι επανάληψης είναι πολύ χρήσιμοι, καθώς επιτρέπουν την επαναλαμβανόμενη εκτέλεση ενός τμήμα κώδικα έως ότου ικανοποιηθούν κάποιες συνθήκες. Οι συνθήκες αυτές μπορεί να περιλαμβάνουν την εισαγωγή δεδομένων από το χρήστη, την αύξηση της τιμής μιας μεταβλητής κλπ. Είναι δυνατόν ο βρόχος να επαναλαμβάνεται εις αεί, εάν δεν έχει περιληφθεί κάποια συνθήκη τερματισμού του.
Υπάρχουν διάφορα είδη βρόχων επανάληψης, όπως: ο βρόχος while, ο do…while, ο for κ.α.

**Ο βρόχος while**

Ο βρόχος "while" είναι ο απλούστερος και πιο συχνά χρησιμοποιημένος στην PHP. Επαναλαμβάνει ένα τμήμα κώδικα εφόσον μια συνθήκη είναι αληθής.
Η γενική μορφή του είναι:

```
while (συνθήκη που είναι αληθής) {
      εντολές προς επανάληψη;
```

```
}
```

Οι εντολές που πρέπει να επαναληφθούν περικλείονται σε αγκύλες. Η κατάσταση της συνθήκης, δηλαδή αν είναι αληθής ή ψευδής, ελέγχεται στην αρχή κάθε επανάληψης του κώδικα.

***Σημείωση**: Μια εναλλακτική δομή του βρόχου είναι και η παρακάτω, όπου δεν χρησιμοποιούνται οι αγκύλες ώστε να περικλείουν τις εντολές προς επανάληψη.*

```
while (συνθήκη):
   εντολές
endwhile;
```

Οι βρόχοι `while` χρησιμοποιούνται συχνά σε συνδυασμό με κάποια ακέραια μεταβλητή, η οποία αυξάνεται ή μειώνεται κατά μία ποσότητα, κάθε φορά που εκτελείται η λούπα. Αυξάνεται (ή μειώνεται) η τιμή της μεταβλητής εντός της επικράτειας του βρόχου. Έτσι σε κάθε επανάληψη ελέγχεται η τιμή της μεταβλητής και αν έχει φτάσει σε κάποια ανώτατη (αντίστοιχα κατώτατη) τιμή, τότε τερματίζεται ο βρόχος. Η ποσότητα κατά την οποία αυξάνεται η μεταβλητή μπορεί να είναι όσες μονάδες απαιτείται. Συνήθως αυξάνεται κατά μία μονάδα.
Έστω ότι χρειάζεται να δημιουργηθεί ένα σενάριο που θα εκτυπώνει αριθμούς από το 1 μέχρι το 10. Τα βήματα που θα ακολουθηθούν είναι τα παρακάτω:

```
α. θέσε τη μεταβλητή $Digit = 1.
β. έλεγξε εάν η τιμή της μεταβλητής είναι ίση ή
μικρότερη από 10.
γ. εκτύπωσε την $Digit.
δ. πρόσθεσε 1 στην $Digit.
ε. πήγαινε στο βήμα β. και εκτέλεσε το σενάριο ξανά
   ελέγχοντας τη νέα τιμή της $Digit.
στ. σταμάτα όταν η $Digit γίνει μεγαλύτερη του 10.
```

Στο παράδειγμα που ακολουθεί μεταφράζεται σε εντολές της γλώσσας PHP το προηγούμενο σενάριο. Πρώτα ορίστηκε μία μεταβλητή και αρχικοποιήθηκε στην τιμή 1. Στην συνέχεια όσο η μεταβλητή παραμένει μικρότερη από 10: (α) εκτυπώνεται και (β) αυξάνεται κατά μία μονάδα. Τελικά, όταν η μεταβλητή γίνει μεγαλύτερη από 10 το πρόγραμμα τερματίζεται.

```
<?php
$Digit = 1;
while ($Digit <= 10)
{
 print ("$Digit");
 $Digit++;
```

```
}
?>
```

Μια παραλλαγή του ανωτέρω κώδικα, παρουσιάζεται και παρακάτω. Οι διαφορές είναι ότι η αύξηση της μεταβλητής τοποθετήθηκε εντός της συνθήκης ελέγχου του βρόχου (`++$Digit <= 10`). Επειδή χρησιμοποιήθηκε ο τελεστής ++ αριστερά σε σχέση με τη μεταβλητή, αυτό είχε ως συνέπεια πρώτα να αυξάνεται η τιμή της και μετά να γίνεται ο έλεγχος με το 10. Επειδή από τον ορισμό του σεναρίου έπρεπε να εκτυπωθούν όλοι οι αριθμοί από το 1 έως 10, για το λόγο αυτό η μεταβλητή αρχικοποιήθηκε στην τιμή 0 (και όχι στην 1 όπως προηγούμενα).

```
<?php
$Digit = 0;
while (++$Digit <= 10)
{
 print ("$Digit");
 }
?>
```

**Ο βρόχος do…while**

Μια διαφοροποίηση του βρόχου `while` αποτελεί ο `do…while`, ο οποίος επαναλαμβάνει την εκτέλεση ορισμένων εντολών έως εφόσον μια συνθήκη είναι αληθής. Η διαφορά τους βρίσκεται στο γεγονός ότι ο βρόχος `do ..while` θα 'τρέξει' τουλάχιστον μια φορά, καθώς η συνθήκη ελέγχεται μετά την εκτέλεση των εντολών.
Η γενική μορφή του είναι:

```
do {
  εντολές προς επανάληψη ;
} while (συνθήκη να είναι αληθής)
```

Στο παράδειγμα που ακολουθεί παρουσιάζεται η χρήση του βρόχου με εντολές της PHP. Το παρακάτω πρόγραμμα εκτυπώνει του αριθμούς από το 1 έως το 10. Σε προηγούμενη ενότητα παρουσιάστηκε το ίδιο σενάριο, αλλά με τη χρήση του βρόχου `while`. Όπως μπορεί κάποιος να παρατηρήσει ο κώδικας δε διαφέρει σημαντικά σε σχέση τη χρήση του βρόχου `do..while` ή του `while`.

```
<?php
$Digit =1;
do{
 print ("$Digit");
 $Digit++;
} while ($Digit <= 10)
?>
```

## Ο βρόχος for

Μια πολυπλοκότερη μορφή βρόχου αποτελεί και ο λούπα `for`. Αυτός ο βρόχος χρησιμοποιείται σε συνδυασμό με μια μεταβλητή που λειτουργεί ως μετρητής των επαναλήψεων. Θεωρείται από πολλούς ο ισχυρότερος βρόχος, καθώς συνδυάζει:

- ✓ τη δυνατότητα της αρχικοποίησης μεταβλητών με την είσοδο στο βρόχο,
- ✓ έλεγχο της συνθήκης κατά την επανάληψη, και
- ✓ διαφοροποίηση μεταβλητών μετά από κάθε γύρο/επανάληψη.

Η γενική μορφή του βρόχου παρουσιάζεται παρακάτω.

```
for (αρχικοποίηση; έλεγχος; μεταβολή) {
    εντολές προς επανάληψη;
}
```

***Σημείωση**: Μια εναλλακτική δομή του βρόχου είναι και η παρακάτω, όπου δεν χρησιμοποιούνται οι αγκύλες ώστε να περικλείουν τις εντολές προς επανάληψη.*
*for (αρχικοποίηση; έλεγχος; μεταβολή ):*
*εντολές*
*endfor;*

Παρακάτω παρουσιάζεται το σενάριο εκτύπωσης αριθμών από το 1 έως το 10, με τη χρήση του βρόγχου `for`. Έχουν οριστεί σε μια γραμμή κώδικα η αρχικοποίηση της μεταβλητής που χρησιμεύει ως μετρητής, ο έλεγχος της συνθήκης που ορίζει τον τερματισμό του βρόχου, καθώς και η μεταβολή του μετρητή. Με το βρόχο `for` είναι δυνατόν να γνωρίζουμε από πριν τον αριθμό των επαναλήψεων που θα πραγματοποιηθούν.

```
<?php
```

```
for ($Digit = 1;$Digit <= 10;$Digit++)
{
 print ("$Digit");
}
?>
```

Αξίζει να επισημανθεί το γεγονός ότι η μεταβλητή μετρητής δεν είναι απαραίτητο να αυτξάνεται κατά τη μεταβολή της. Μπορεί, για παράδειγμα, να μειώνεται, όπως φαίνεται στον παρακάτω κώδικα όπου εκτυπώνει αντίστροφα τους αριθμούς από το 1 έως το 10.

```
<?php
for ($Digit = 10;$Digit >= 0;$Digit--)
{
 print ("$Digit");
}
?>
```

Εάν δε δοθούν συνθήκες στο βρόχο αυτός θα επαναλαμβάνεται ατέρμονα, καθώς η PHP εξ ορισμού την έλλειψη συνθηκών την λαμβάνει ως αληθής. Για παράδειγμα το παρακάτω θα επαναλαμβάνεται χωρίς τέλος.

```
<?php
for (;;) {
 print ("Ατέρμονο λουπ");
}
?>
```

**Ο βρόχος foreach**

Μια ειδική μορφή βρόχου είναι ο foreach, ο οποίος μπορεί να εφαρμοστεί μόνο σε πίνακες και αντικείμενα. Μπορεί να επαναλάβει την εκτέλεση ορισμένων

εντολών τόσες φορές όσες είναι τα στοιχεία του πίνακα. Είναι εξαιρετικά χρήσιμος στις περιπτώσεις όπου χρειάζεται να πραγματοποιηθούν ενέργειες για κάθε στοιχείο ενός πίνακα.
Η γενική μορφή του βρόχου παρουσιάζεται παρακάτω.

```
foreach (όνομα_πίνακα as τιμή_μεταβλητή) {
    εντολές προς επανάληψη;
}
```

Παρακάτω παρουσιάζεται η χρήση του βρόχου στην εκτύπωση των στοιχείων ενός απλού πίνακα.

```
<?php
  $students = array ( "Ανδριαννή", "Ηλλιάνα", "Λευτέρης",
"Γιάννης" );
  foreach ($students as $each_value) {
    echo $each_value;
  }
?>
```

Το πρώτο στοιχείο του πίνακα βρίσκεται στη θέση 0, δηλαδή είναι το `$students[0]`. Κατά την πρώτη επανάληψη του βρόχου, η τιμή του πρώτου στοιχείου του πίνακα (Ανδριαννή) αποθηκεύεται στη μεταβλητή `$each_value`. Δηλαδή πραγματοποιείται το: `$each_value= $students[0]`. Μέσα στο σώμα του βρόχου, στις εντολές που επαναλαμβάνονται, χρησιμοποιείται μόνο η μεταβλητή ως αναφορά στην τιμή του στοιχείου του πίνακα. Στη συνέχεια, κατά τη δεύτερη επανάληψη του βρόχου, στη μεταβλητή `$each_value` θα εκχωρηθεί η τιμή του δεύτερου στοιχείου του πίνακα, δηλαδή του `$students[1]` κ.ο.κ. Η διαδικασία θα επαναληφθεί έως ότου εξαντληθούν τα στοιχεία του πίνακα.

Μέχρι τώρα παρουσιάστηκε ο τρόπος πρόσβασης στη τιμή κάθε στοιχείου, αλλά όχι και στο κλειδί του. Υπάρχει η πιθανότητα να χρειάζεται να πραγματοποιηθούν ενέργειες ανάλογα τη θέση (κλειδί) κάθε στοιχείου ενός πίνακα. Στην περίπτωση που ο βρόχος `foreach` χρησιμοποιείται στα στοιχεία συσχετιζόμενου πίνακα έχει η γενική μορφή:

```
foreach(πίνακας as κλειδί_μεταβλητή=>τιμή_μεταβλητή) {
    εντολές προς επανάληψη;
}
```

Όπως φαίνεται η σύνδεση της μεταβλητής που αποθηκεύεται το κλειδί και της μεταβλητής που περιέχει τη τιμή του στοιχείου του πίνακα, πραγματοποιείται με το σύμβολο =>. Παρακάτω παρουσιάζεται ένα παράδειγμα σε κώδικα PHP της χρήσης του βρόχου στην περίπτωση συσχετιζόμενου πίνακα.

```
<?php
$characters = array ( "name"=>"Τόνυ",
                      "haircolor"=>"μαύρα",
                      "eyecolor"=>"καστανά",
                      "age"=>5);

  foreach ($characters as $each_key => $each_value) {
    echo $each_key. " είναι ".$each_value. "\n";
  }
?>
```

***Σημείωση**: Μια εναλλακτική δομή του βρόχου είναι και η παρακάτω, όπου δεν χρησιμοποιούνται οι αγκύλες ώστε να περικλείουν τις εντολές προς επανάληψη.*
*foreach (συνθήκη):*
*εντολές*
*endforeach;*

**Διακοπή, υπερπήδηση και ανακατεύθυνση**

Υπάρχουν φορές όπου χρειάζεται να διαφοροποιηθεί η κανονική λειτουργία ενός βρόχου. Κάτω από ορισμένες συνθήκες, μπορεί, να χρειάζεται να διακοπούν οι επαναλήψεις, ή να παραλειφθεί κάποια από αυτές είτε να ανακατευθυνθεί η ροή του κώδικα προς συγκεκριμένο τμήμα. Οι ανωτέρω ενέργειες πραγματοποιούνται με τις εντολές: `break`, `continue` και `goto`.

***break***

Η εντολή break προκαλεί τη διακοπή της εκτέλεσης του βρόχου που την περιέχει και όλων των εσωτερικών προς αυτήν. Σε προηγούμενη ενότητα παρουσιάστηκε η χρήση της σε σχέση με την εντολή διακλάδωσης switch. Η δυνατότητα διακοπής της εκτέλεσης ενός βρόχου αποδεικνύεται εξαιρετικά χρήσιμη στην

περίπτωση εμφάνισης κάποιου λάθους στο πρόγραμμα. Συνήθως χρησιμοποιείται σε συνδυασμό με κάποια εντολή ελέγχου συνθήκης (π.χ. if).
Για παράδειγμα το παρακάτω σενάριο διακόπτει την εκτέλεση του βρόχου εφόσον υπάρχει βρεθεί η λέξη stop.

```
<?php
$arr = array('ένα', 'δύο', 'τρία', 'τέσσερα', 'stop',
'πέντε');
foreach ($arr as $val) {
    if ($val == 'stop') {
        break;        }
    echo "$val ";
}
?>
```

Το αποτέλεσμα θα είναι το : ένα δύο τρία τέσσερα. Δεν εκτυπώθηκε το λεκτικό "πέντε", καθώς βρέθηκε αληθής η συνθήκη $val == 'stop' και εκτελέστηκε η εντολή break.

***Σημείωση**: Εάν χρειάζεται να εκτυπωθεί μια κενή γραμμή, αυτό γίνεται με τη χρήση του χαρακτήρα διαφυγής \ και το λεκτικό n (new line) ως εξής: echo "μπλα μπλα \n".*

Είναι δυνατόν η break να δεχτεί αριθμητικό όρισμα. Το αριθμητικό όρισμα της καθορίζει πόσους εξωτερικούς βρόχους θα διακόψει. Στο προηγούμενο παράδειγμα θα μπορούσε η εντολή break; να αντικατασταθεί από την break 1;, που σημαίνει "διέκοψε ένα (1) βρόχο που περιέχει την εντολή break". Είναι δυνατόν να υπάρχουν εμφωλευμένοι βρόχοι. Σε αυτή την περίπτωση η break μπορεί, με το κατάλληλο όρισμα, να προκαλέσει τη διακοπή όλων των εξωτερικών επιπέδων των βρόχων. Για παράδειγμα το παρακάτω:

```
<?php
$arr = array('ένα', 'δύο', 'τρία', 'τέσσερα', 'πέντε');
foreach ($arr as $val) {
    echo "$val \n";
    switch ($val) {
        case 'δύο':
```

```
            echo "Έξοδος από το switch \n";
            break 1;
        case 'τέσσερα':
            echo "Έξοδος και από το βρόχο \n";
            break 2;
        }
}
?>
```

Έχει ως αποτέλεσμα το:

```
ένα
δύο
Έξοδος από το switch
τρία
τέσσερα
Έξοδος και από το βρόχο
```

***continue***

Στην περίπτωση όπου χρειάζεται να παραληφθεί κάποιος κύκλος επανάληψης ενός βρόχου, τότε χρησιμοποιείται η εντολή continue. Η εντολή continue μοιάζει αρκετά με τη break. Η διαφορά τους είναι ότι η continue θα προκαλέσει τη διακοπή/υπερπήδηση μόνο της τρέχουσας επανάληψης. Ο βρόχος θα συνεχίσει κανονικά στην επόμενη επανάληψη. Αυτού του είδους η συμπεριφορά είναι βολική καθώς μπορεί να οδηγήσει σε εξοικονόμηση υπολογιστικών πόρων ή την αποφυγή κάποιου λάθους. Για παράδειγμα ο παρακάτω κώδικας θα διακόψει τον κύκλο της επανάληψης όπου πρόκειται να πραγματοποιηθεί διαίρεση με 0.

```
<?php
$j = 10;
while ($j > -10) {
      $j--;
      if ($j == 0) continue;
      echo (10 / $j)."\n";
```

```
}
?>
```

Η continue, όπως και η break, είναι δυνατόν να δεχτεί αριθμητικό όρισμα. Το οποίο καθορίζει σε πόσους εξωτερικούς βρόχους θα διακόψει τον τρέχοντα κύκλο επανάληψης.

***goto***

Η εντολή goto μπορεί να χρησιμοποιηθεί ώστε να ανακατευθυνθεί η ροή του προγράμματος προς συγκεκριμένο τμήμα του κώδικα. Τα τμήμα του κώδικα που αποτελεί τον προορισμό ορίζεται από μια ετικέτα (label), σύμφωνα με τη μορφή: "ετικέτα:". Η ετικέτα μπορεί να είναι όποιο λεκτικό αποφασίσει ο προγραμματιστής. Η ανακατεύθυνση πραγματοποιείται με τη χρήση της goto σε συνδυασμό με την ετικέτα, δηλαδή goto ετικέτα. Για παράδειγμα ο παρακάτω κώδικας παρουσιάζει τη χρήση της goto, ως εντολή εξόδου από βρόχο.

```
<?php
$j=50;
for($i=0; $i<100; $i++) {
  while($j--) {
    if($j==17) goto end;
  }
}
echo "i = $i";
end:
echo 'Το j έγινε 17';
?>
```

Η ανακατεύθυνση είναι χρήσιμη στην περίπτωση όπου χρειάζεται να πραγματοποιηθεί έξοδος από πολλαπλούς εμφωλευμένους βρόχους.
Η χρήση της υπόκειται σε περιορισμούς. Ο προορισμός θα πρέπει να είναι μέσα στο ίδιο αρχείο και στο ίδιο ‘περιεχόμενο’. Δηλαδή, δε γίνεται να χρησιμοποιηθεί για την έξοδο από κάποια συνάρτηση ή μέθοδο. Επίσης ο προορισμός δε μπορεί να είναι εντός κάποιου βρόχου ή διακλάδωσης switch.
Αξίζει να τονιστεί ότι οι εντολές που ακολουθούν την ετικέτα προορισμού, θα εκτελεστούν χωρίς περιορισμούς. Δηλαδή δε πρόκειται για κάποιο κλειστό τμήμα κώδικα που εκτελείται μόνο εάν μεταφερθεί σε αυτό ο έλεγχος μέσω της goto. Η ετικέτα προορισμού απλά σημειώνει ένα σημείο στη ροή του

προγράμματος όπου μπορεί να φτάσει κάποιος ακολουθώντας ένα διαφορετικό μονοπάτι.

### 5.2.10 Συναρτήσεις στην PHP

Οι συναρτήσεις περιλαμβάνουν τμήματα κώδικα, τα οποία χρησιμοποιούνται σε διαφορετικά σημεία μέσα σε ένα αρχείο. Μερικά από τα πλεονεκτήματα χρήσης των συναρτήσεων είναι:

- ✓ Μειώνουν τις επαναλήψεις κώδικα — Οι συναρτήσεις επιτρέπουν την οργάνωση επαναλαμβανόμενων τμημάτων κώδικα σε ένα μοναδικό στοιχείο. Η εκτέλεση της ίδιας εργασίας μπορεί να πραγματοποιηθεί με απλή κλήση της συνάρτησης, χωρίς να απαιτείται η αντιγραφή και επικόλληση τμημάτων κώδικά. Εξοικονομείται έτσι χρόνος και κόπος.
- ✓ Διευκολύνουν τη συντήρηση του προγράμματος — Μια συνάρτηση δημιουργείται μια φορά και χρησιμοποιείται σε πολλαπλά σημεία. Η τυχόν αλλαγές πραγματοποιούνται μόνο στον κώδικά της συνάρτησης, αλλά έχουν αντίκτυπο σε πολλαπλά σημεία του προγράμματος.
- ✓ Διευκολύνουν την επίλυση λαθών — Ο κώδικας γίνεται ευανάγνωστος και ευπαρουσίαστος, οπότε και η αναγνώριση των τμημάτων που παρουσιάζουν λάθη γίνεται ευκολότερη.
- ✓ Μπορούν να χρησιμοποιηθούν από διαφορετικές εφαρμογές — Αν η συνάρτηση οριστεί σε ξεχωριστό αρχείο μπορεί να κληροδοτηθεί σε διαφορετικές εφαρμογές. Για να υλοποιηθεί αυτό αρκεί να συμπεριληφθεί το αρχείο που περιέχει τον ορισμό της (χρήση των εντολών include, require κλπ) στο αρχείο όπου γίνεται κλήση της.

Ανάλογα με το δημιουργό τους, υπάρχουν συναρτήσεις που έχει ορίσει ο προγραμματιστής (user defined) και άλλες οι οποίες είναι ενσωματωμένες (build-in) στη γλώσσα PHP. Μερικές χρειάζονται κάποιες παραμέτρους για να λειτουργήσουν, ενώ άλλες όχι. Ορισμένες με την εκτέλεσή τους επιστρέφουν κάποια τιμή ως αποτέλεσμα, ενώ άλλες πραγματοποιούν μια ενέργεια.

**<u>Ορισμός συνάρτησης</u>**

Η PHP είναι μία εξαιρετικά ευέλικτη γλώσσα σεναρίων, η οποία παρέχει πολλές δυνατότητες στη δημιουργία συναρτήσεων. Οι συναρτήσεις που δημιουργεί ο προγραμματιστής εκτελούνται με τον ίδιο τρόπο με τις συναρτήσεις ενσωματωμένες της PHP και έχουν την μορφή:

```
<?php
function MyFunction ()
{
```

```
//κώδικας συνάρτησης
}
?>
```

Ο ορισμός μιας συνάρτησης γίνεται με τη φράση: function …….. ( ). Το λεκτικό: MyFunction αποτελεί το όνομα της συνάρτησης. Στις αγκύλες που ακολουθούν ορίζεται ο κώδικα που περιέχει η συνάρτηση. Η ονοματολογία των συναρτήσεων ακολουθεί τους ίδιους κανόνες με αυτή των μεταβλητών, με τη διαφορά ότι δεν αρχίζει με το σύμβολο του $ και δεν υπάρχει διαχωρισμός πεζών και κεφαλαίων (case insensitive). Το όνομα πρέπει να γράφεται χωρίς κενά και να αρχίζει από γράμμα ή την κάτω παύλα.

***Σημείωση**: Αποτελεί καλή πρακτική τα ονόματα των συναρτήσεων να υποδηλώνουν και τη λειτουργία τους.*

Μια απλή συνάρτηση παρουσιάζεται παρακάτω.

```
<?php
function MyMessage(){
   print "Αυτό είναι ένα απλό παράδειγμα συνάρτησης";
}
?>
```

Η δήλωση `function MyMessage ()` ορίζει τη συνάρτηση με όνομα `MyMessage`. Το περιεχόμενο της συνάρτησης, το οποίο περικλείεται σε άγκιστρα { }, εκτυπώνει τη φράση "`Αυτό είναι ένα απλό παράδειγμα συνάρτησης`".

## Κλήση συνάρτησης

Μια συνάρτηση δεν εκτελείται εάν πρώτα δεν κληθεί. Η κλήση των συναρτήσεων είναι ίδια ανεξάρτητα το δημιουργό τους. Στο ανωτέρω παράδειγμα η συνάρτηση θα κληθεί ως: `MyMessage ();`.

Η PHP εφόσον μετά το όνομα της μεταβλητής ακολουθούν παρενθέσεις, προσπαθεί να βρει κάποια συνάρτηση με αυτό το όνομα. Αν σε κάποια μεταβλητή δοθεί ως τιμή το όνομα μιας συνάρτησης τότε η PHP θα την τρέξει/ εκτελέσει κανονικά. Αυτό ονομάζεται μεταβλητή συνάρτηση (variable function). Για παράδειγμα το παρακάτω σενάριο αρχικά εκχωρεί τιμή στη μεταβλητή `$temp` και στη συνέχεια καλείται η συνάρτηση με το ίδιο όνομα.

```
<?php
      function foo() {
```

```
    echo "In foo() \n";
}
$temp = 'foo';
$temp();         // κλήση της foo()
?>
```

**Παράμετροι**

Η γλώσσα php παρέχει τη δυνατότητα να χρησιμοποιηθούν παράμετροι, ώστε να δοθούν πληροφορίες τις οποίες χρειάζεται η συνάρτηση για τη λειτουργία της. Οι παράμετροι διαχωρίζονται με κόμμα και δεν υπάρχει περιορισμός στον αριθμό τους. Η PHP δίνει τιμές στις παραμέτρους από αριστερά προς τα δεξιά.
Αυτές οι παράμετροι λειτουργούν ως μεταβλητές στο εσωτερικό της συνάρτησης.

***Σημείωση**: Το όρισμα είναι μια τιμή που παρέχεται στη συνάρτηση και παράμετρος είναι η μεταβλητή εντός της συνάρτησης στην οποία εκχωρείται η τιμή αυτή. Χρησιμοποιούνται πανομοιότυπα.*

Υπάρχουν συναρτήσεις που απαιτούν παραμέτρους και άλλες που δεν απαιτούν. Το παρακάτω παράδειγμα δέχεται ως παραμέτρους δυο ακέραιους και εκτυπώνει το άθροισμά τους.

```
<?php
        function MySum($num1, $num2) {
           $sum = $num1 + $num2;
           echo "Το άθροισμα είναι: $sum";
        }

        MySum(10, 20); //άθροισμα =30
?>
```

Είναι δυνατόν να οριστούν προκαθορισμένες τιμές για τις παραμέτρους, ώστε να αποφευχθούν τυχόν παραλείψεις. Έτσι το προηγούμενο παράδειγμα γίνεται:

```
<?php
```

```
        function MySum($num1, $num2=30) {
            $sum = $num1 + $num2;
            echo "Το άθροισμα είναι: $sum";
        }
        MySum(10, 20); // άθροισμα = 30
        MySum(10); //άθροισμα =40
?>
```

Στη δεύτερη κλήση της συνάρτησης έχει παραληφθεί το δεύτερο όρισμα. Εντούτοις δεν υπάρχει λάθος, καθώς χρησιμοποιείται η προκαθορισμένη τιμή (30). Οι παράμετροι που θα έχουν προκαθορισμένες τιμές θα πρέπει να είναι τελευταίες στη λίστα με τις παραμέτρους του ορισμού της συνάρτησης. Για παράδειγμα ο ορισμός: function MySum($num1=30, $num2), δε θα δώσει τα αποτελέσματα που αναμένονται. Το σωστό είναι: function MySum($num1, $num2=30).

Ενδιαφέρον παρουσιάζει η περίπτωση όπου η λίστα με τις παραμέτρους έχει μεταβλητό αριθμό στοιχείων. Από την έκδοση 5.6+ της PHP, αυτό δηλώνεται με τη χρήση του συμβόλου '…' στον ορισμό της συνάρτησης. Ο συμβολισμός αυτός μετατρέπει την παράμετρο σε πίνακα μεταβλητού μήκους. Έτσι το παράδειγμα του αθροίσματος ακεραίων γίνεται:

```
<?php
        function MySum(...$num) {
            $sum=0;
            foreach ($num as $n) {
                $sum += $n;
            }
            echo "Το άθροισμα είναι: $sum";
        }
          MySum(10, 20); //άθροισμα =30
          MySum(10, 20,40); //άθροισμα =70
          MySum(1, 2, 3, 4); //άθροισμα =10
?>
```

Εντός της συνάρτησης η παράμετρος $num συμπεριφέρθηκε ως πίνακας.

Εξ ορισμού οι παράμετροι στις συναρτήσεις εκχωρούνται κατά τιμή (by value). Αυτό σημαίνει ότι αν η τιμή της παραμέτρου αλλάξει εντός της συνάρτησης, η αλλαγή δεν ισχύει και έξω από αυτήν. Για να διατηρηθεί η αλλαγή και εκτός της συνάρτησης θα πρέπει οι παράμετροι να εκχωρηθούν κατά αναφορά (by reference). Όλα τα προηγούμενα παραδείγματα καλούσαν κάποια συνάρτηση με αριθμητικά ορίσματα. Η εκχώρηση με αναφορά έχει νόημα όταν κατά την κλήση παρέχεται κάποια μεταβλητή, όπως το παρακάτω παράδειγμα.

```
<?php
          function MySum(&$num1, $num2) {
             $num1 += $num2;
             echo "Το άθροισμα είναι: $num1";
          }
          $n = 10;
          MySum($n, 20); //άθροισμα =30
         echo "Εκτός συνάρτησης είναι: $n"; //δίνει 30
?>
```

Η εκχώρηση κατά αναφορά πραγματοποιείται με τη χρήση του συμβόλου & πριν το όνομα της μεταβλητής, π.χ. &$num1. Η κλήση έγινε με όρισμα τη μεταβλητή $n, της οποίας αυξήθηκε η τιμή και διατήρησε την αλλαγή και μετά το πέρας της συνάρτησης.

**Επιστροφή**

Οι συναρτήσεις μπορούν να επιστρέψουν τιμές με τη χρήση της εντολής return. Κατά τη εκτέλεση της συνάρτησης, αμέσως μόλις απαντηθεί μια εντολή return η συνάρτηση θα πρέπει να επιστρέψει τον έλεγχο πίσω στο πρόγραμμα και συγκεκριμένα στο σημείο από όπου κλήθηκε. Είναι δυνατόν να επιστραφεί οποιοσδήποτε τύπος δεδομένων, συμπεριλαμβανομένου των πινάκων και των αντικειμένων.
Το παρακάτω παράδειγμα επιστρέφει το άθροισμα δύο ακεραίων.

```
<?php
      function MySum($num1, $num2) {
            return $num1 + $num2;
      }
```

```
      echo "Το άθροισμα είναι:". MySum(10, 20);
      $temp= MySum(10, 20); //αποθήκευση σε μεταβλητή
?>
```

Η τιμή που επιστρέφεται μπορεί να αποθηκευτεί σε μια μεταβλητή, όπως παρουσιάζεται στο προηγούμενο παράδειγμα.
Μια συνάρτηση δεν μπορεί να επιστρέψει πολλαπλές τιμές. Εναλλακτικά μπορεί να επιστραφεί κάποιος πίνακας. Για παράδειγμα το παρακάτω σενάριο επιστρέφει το άθροισμα και τη διαφορά δυο ακεραίων ως στοιχεία ενός πίνακα.

```
<?php
      function MySum($num1, $num2) {

            return array ($num1 + $num2,$num1 - $num2 );
      }
      print_r(MySum(10, 20));
      $temp= MySum(10, 20); //αποθήκευση σε μεταβλητή
?>
```

Το αποτέλεσμα της εντολής print_r(MySum(10, 20)); θα είναι το:
Array ( [0] => 30 [1] => -10 )

### 5.2.11 Εμβέλεια μεταβλητών

Εμβέλεια είναι το πεδίο/χώρος/περιεχόμενο μέσα στο οποίο ορίζεται μια μεταβλητή και μπορεί να χρησιμοποιηθεί. Κάθε μεταβλητή έχει ένα συγκεκριμένο πλαίσιο μέσα στο οποίο είναι έγκυρη και μόνο κώδικας μέσα στο ίδιο πεδίο έχει πρόσβαση στη μεταβλητή.
Η χρήση και κατανόηση της εμβέλειας των μεταβλητών είναι ωφέλιμη στη συγγραφή πολύπλοκων εφαρμογών. Εάν όλες οι μεταβλητές ήταν διαθέσιμες παντού σε μια εφαρμογή, τότε η ανίχνευση και διόρθωση των λαθών θα ήταν μια εξαιρετικά δύσκολη υπόθεση. Επιπλέον η εύρεση μοναδικών σύντομων ονομάτων, που φανερώνουν το σκοπό της ύπαρξης τους, για όλες αυτές τις μεταβλητές θα ήταν δυσχερής διαδικασία.
Ανάλογα με την εμβέλεια τους οι μεταβλητές μπορεί να χαρακτηριστούν ως:

- ✓ Τοπικές -- Οι τοπικές μεταβλητές είναι προσβάσιμες μόνο από το τμήμα του κώδικα, όπου έχουν οριστεί. Αν βρίσκονται έξω από μια συνάρτηση, τότε μπορεί να προσπελαστούν από όλο τον κώδικα που βρίσκεται έξω από συναρτήσεις, τάξεις (classes), κ.ο.κ. Αντιθέτως, αν μια μεταβλητή έχει οριστεί εντός μιας συνάρτησης, μόνο ο κώδικας της συνάρτησης μπορεί να έχει πρόσβαση στη μεταβλητή. Η τιμή της θα χαθεί, όταν η συνάρτηση ολοκληρώνεται.
- ✓ Στατικές – Οι στατικές μεταβλητές μοιάζουν αρκετά με τις τοπικές. Για τον ορισμό τους χρησιμοποιείται η εντολή static. Έχουν εμβέλεια μόνο μέσα στη συνάρτηση όπου έχουν οριστεί, αλλά δε χάνουν την τιμή τους όταν ολοκληρωθεί η συνάρτηση. Αρχικοποιούνται μόνο μια φορά και συγκεκριμένα την πρώτη φορά που θα 'τρέξει' η συνάρτηση. Μπορεί να χρησιμοποιηθούν πολλές φορές μέσα στην ίδια συνάρτηση, διατηρώντας την τιμή τους ανάμεσα στις διαφορετικές κλήσεις.
- ✓ Καθολικές – Οι καθολικές μεταβλητές μπορεί να προσπεραστούν από οποιοδήποτε τμήμα του κώδικα της εφαρμογής.

**Τοπικές μεταβλητές**

Τοπικές είναι οι μεταβλητές που έχουν οριστεί μέσα και μπορούν να προσπελαστούν μόνο μέσα από μια συνάρτηση. Γενικά πρόκειται για προσωρινές μεταβλητές, οι οποίες χρησιμοποιούνται ώστε να συλλεγούν αποτελέσματα υπολογισμών εντός της συνάρτησης. Ένα παράδειγμα τέτοιων μεταβλητών αποτελούν και οι παράμετροι των συναρτήσεων. Οι παράμετροι αυτοί ως μεταβλητές έχουν χρήση μόνο από τη συνάρτηση, δηλαδή δεν μπορεί να τους εκχωρηθεί κάποια τιμή ή να πάρουν μέρος σε κάποιο υπολογισμό εκτός του 'σώματος' της συνάρτησης. Σε προηγούμενη ενότητα, όταν εξεταζόταν οι συναρτήσεις, παρουσιάστηκε το παρακάτω παράδειγμα.

```
<?php
      function MySum($num1, $num2) {
            $sum = $num1 + $num2;
            echo "Το άθροισμα είναι: $sum";
          }

          MySum(10, 20); //άθροισμα =30
```

```
?>
```

Οι παράμετροι της συνάρτησης, `$num1, $num2`, αποτελούν τοπικές μεταβλητές. Εντός της συνάρτησης χρησιμοποιήθηκε η μεταβλητή `$sum`, ως αποδέκτης υπολογισμών, η οποία, επίσης, είναι τοπική μεταβλητή.

***Σημείωση**: Εξ ορισμού οι μεταβλητές που ορίζονται εντός μιας συνάρτησης είναι τοπικές.*

Αποτελεί σύνηθες προγραμματιστικό λάθος η παράλειψη (και παράβλεψη) της εμβέλειας των μεταβλητών, η οποία οδηγεί στη χρονοβόρα διαδικασία εντοπισμού σφαλμάτων στον κώδικα.

**Στατικές μεταβλητές**

Μια άλλη κατηγορία μεταβλητών είναι και οι στατικές. Μοιάζουν αρκετά με τις τοπικές, αλλά διαφέρουν στο γεγονός ότι η τιμή τους δε μηδενίζεται μόλις τερματιστεί η εκτέλεση της συνάρτησης που τις περιέχει. Μια μεταβλητή ορίζεται ως στατική με τη χρήση της λέξης κλειδί `static`. Στις νεότερες εκδόσεις της PHP μπορεί να εκχωρηθεί τιμή απευθείας με τον ορισμό της μεταβλητής. Εντούτοις η τιμή αυτή δεν μπορεί να προέρχεται από κλήση άλλης συνάρτησης. Στο παρακάτω παράδειγμα παρουσιάζονται σωστοί, αλλά και λάθος, τρόποι ορισμού μιας στατικής μεταβλητής.

```
<?php
function some_function(){
      static $int = 0;       // σωστό
      static $int = 1+3;     // σωστό (ισχύει για PHP 5.6+)
      static $int = max(1,2,3);  // λάθος


      /*κώδικας συνάρτησης*/
}
?>
```

Στο επόμενο παράδειγμα παρουσιάζεται η χρήση στατικών μεταβλητών στο σενάριο όπου χρειάζεται να καταγραφεί πόσες φορές έχει κληθεί μια συνάρτηση.

```
<?php
function my_static()
{
      static $count = 0;
```

```
      echo $count;
      $count++;
}
?>
```

Η πρώτη γραμμή κώδικα της συνάρτησης ορίζει μια στατική μεταβλητή, που ονομάζεται `$count` και της εκχωρεί την τιμή 0. Στην επόμενη γραμμή εμφανίζει την τιμή της μεταβλητής και τελικά την αυξάνει, κάνοντάς της 1.
Την επόμενη φορά που καλείται η συνάρτηση, επειδή η `$count` έχει ήδη οριστεί, παραλείπεται η πρώτη γραμμή κώδικα της συνάρτησης. Η μεταβλητή `$count` θα έχει την τιμή της τελευταίας κλήσης, που στην αυτήν την περίπτωση είναι το 1. Στη συνέχεια παρουσιάζεται αυτή η τιμή και μετά αυξάνεται κατά μία μονάδα, οπότε γίνεται 2. Αυτή η διαδικασία θα επαναληφθεί όσες φορές κληθεί η συνάρτηση.
Οι στατικές μεταβλητές βρίσκουν εφαρμογή στον ορισμό των αναδρομικών συναρτήσεων. Αναδρομική είναι μια συνάρτηση που καλεί τον εαυτό της. Χρειάζεται προσοχή ώστε να διασφαλιστεί ο τερματισμός της επαναλαμβανόμενης εκτέλεσης της συνάρτησης, διαφορετικά μπορεί να σχηματιστεί ατέρμονή λούπα. Ένα παράδειγμα αναδρομικής συνάρτησης είναι το παρακάτω.

```
<?php
function my_static ()
{
      static $count = 0;

      $count++;
      echo $count;
      if ($count < 10) {
            my_static ();
      }
      $count--;
}
my_static();
```

```
?>
```

Αρχικά ορίζεται και αρχικοποιείται η στατική μεταβλητή `$count`. Στη συνέχεια αυτή αυξάνεται και εκτυπώνεται στην οθόνη. Η συνάρτηση καλεί αναδρομικά τον εαυτό της εάν η μεταβλητή έχει τιμή μικρότερη από 10. Στη συνέχεια η μεταβλητή μειώνεται ώστε η συνάρτηση να μπορεί να εκτελεστεί και πάλι. Διαφορετικά αν η μεταβλητή `$count` δε μειωνόταν μετά την πρώτη εκτέλεση θα ήταν πάντα 10 και η συνάρτηση δε θα μπορούσε να εκτελεστεί επιπλέον.

**Καθολικές μεταβλητές**

Υπάρχουν περιπτώσεις όπου χρειάζεται μια μεταβλητή να είναι καθολική, δηλαδή να μπορεί να προσπελαστεί από οποιοδήποτε τμήμα κώδικα, είτε αυτό είναι μέσα σε συνάρτηση/αντικείμενο/τάξη είτε έξω. Επιπλέον, ορισμένες δομές δεδομένων μπορεί να είναι πολύπλοκες, οπότε να δυσχεραίνεται η χρήση τους ως παραμέτροι συναρτήσεων. Αυτές είναι μερικές από τις περιπτώσεις όπου μια μεταβλητή χρειάζεται να οριστεί ως καθολική (global).

Για να οριστεί μια μεταβλητή ως καθολική χρησιμοποιείται η λέξη κλειδί `global`. Με αυτόν τον τρόπο, χρησιμοποιώντας δηλαδή την λέξη `global`, δε μπορεί να γίνει και ταυτόχρονη αρχικοποίηση της μεταβλητής. Η μεταβλητή θα πρέπει να αρχικοποιηθεί σε διαφορετική εντολή.

Στο παρακάτω παράδειγμα παρουσιάζονται σωστοί, αλλά και λάθος, τρόποι ορισμού μιας καθολικής μεταβλητής.

```
<?php
function some_function(){
      global $int;
      $int = 0;                 //σωστό
      global $int; $int = 0; //σωστό
      global $int = 0;       //λάθος

      $GLOBALS["var"]=0;//σωστό ορισμός και αρχικοποίηση
      $GLOBALS["int"]=99;   //εκχώρηση τιμής

      echo $int;              //εκτυπώνει 99
}
?>
```

Ένας διαφορετικός τρόπος ορισμού μιας καθολικής μεταβλητής είναι να δηλωθεί ως στοιχείο του πίνακα `$GLOBAL`. Ο πίνακας αυτός περιέχει ως στοιχεία του όλες τις μεταβλητές της εφαρμογής που έχουν, με οποιονδήποτε, τρόπο οριστεί ως καθολικές. Έτσι μπορεί να οριστεί και να αρχικοποιηθεί μια καθολική μεταβλητή. Το όνομα της μεταβλητής αποτελεί και το κλειδί/θέση της στον πίνακα.
Στο ανωτέρω παράδειγμα το αποτέλεσμα της `echo` θα είναι 99, καθώς αυτή η τιμή έχει δοθεί στην καθολική μεταβλητή `$int`, προελαύνοντας τη θέση της στον πίνακα `$GLOBALS`.

***Σημείωση**: Η χρήση της λέξης κλειδί global εκτός του σώματος μιας συνάρτησης δεν αποτελεί λάθος, απλά δεν έχει ιδιαίτερο νόημα αφού εξ ορισμού όλες οι μεταβλητές εντός σεναρίου και εκτός συναρτήσεων είναι καθολικές.*

Το παρακάτω αποτελεί σύνηθες παράδειγμα εφαρμογής των καθολικών μεταβλητών. Έστω ότι χρειάζεται να αυθεντικοποιηθούν οι χρήστες ενός ιστοτόπου. Επιπλέον θα πρέπει όλα τα τμήματα κώδικα της εφαρμογής να καταλαβαίνουν ότι πρόκειται για επισκέπτη που έχει συνδεθεί στον ιστοτόπο. Ένας τρόπος να 'περαστεί' αυτή η πληροφορία σε όλα τα τμήματα της εφαρμογής είναι να οριστεί μια μεταβλητή ως καθολική, ως εξής:

```
global logged_in;
```

Οπότε όταν ο χρήσης συνδέεται στη μεταβλητή τίθεται η τιμή 1 και όποτε αποσυνδέεται η 0. Σε αυτή την περίπτωση κάθε τμήμα της εφαρμογής γνωρίζει την τιμή της, χρησιμοποιώντας έναν απλό έλεγχο, όπως:

```
if ($logged_in==1) {/*κάνε κάτι */}
```

***Σημείωση**: Μια πρακτική αποτελεί η χρήση μόνο κεφαλαίων γραμμάτων στην ονοματολογία των καθολικών μεταβλητών, οπότε άμεσα φανερώνεται η εμβέλεια τους.*

# Αυτά για την ασφάλεια είναι περισσότερο για super global ή μήπως όχι;

Η χρήση των καθολικών μεταβλητών κάνει τη συντήρηση του κώδικα δυσκολότερη, ιδιαίτερα με την πάροδο του χρόνου.

- ✓ Μια καθολική μεταβλητή μπορεί να οριστεί οπουδήποτε, ή πουθενά, οπότε δεν υπάρχει κάποιο ειδικό τμήμα του κώδικά που να περιέχει επεξηγηματικά σχόλια για τη χρήση της.
- ✓ Κατά κανόνα όταν διαβάζεται ο κώδικας, ενστικτωδώς, υποτίθεται ότι οι μεταβλητές είναι τοπικές οπότε μια αλλαγή της μεταβλητής μέσα σε μια συνάρτηση μπορεί να φέρει αλλαγή σε όλη την εφαρμογή.
- ✓ Οι συναρτήσεις επιστρέφουν τις ίδιες τιμές εφόσον χρησιμοποιούνται οι ίδιες παράμετροι. Η χρήση καθολικών μεταβλητών εντός της συνάρτησης αλλάζει αυτό το αποτέλεσμα, χωρίς να είναι προφανές από τον ορισμό της συνάρτησης.
- ✓ Επειδή οι καθολικές μεταβλητές δεν έχουν κάποιο συγκεκριμένο τρόπο αρχικοποίησης, δεν είναι ποτέ σίγουρο αν τους έχει εκχωρηθεί κάποια τιμή.
- ✓ Κάποια διαφορετική εφαρμογή (ίσως κάποιο plugin) μπορεί να χρησιμοποιεί καθολικές μεταβλητές με το ίδιο όνομα, με αποτέλεσμα να επηρεάζεται η λειτουργία της εφαρμογής που ξεκινά δεύτερη (το plugin ή η δική σας).

Η χρήση καθολικών μεταβλητών θα πρέπει να γίνεται με φειδώ. Αποτελεί καλή πρακτική η αποφυγή της χρήσης τους. Οι λόγοι που ωθούν προς αυτή την κατεύθυνση δεν είναι μόνον πρακτικοί, όπως η οικονομία ονομάτων ή δυσκολία στην αποσφαλμάτωση της εφαρμογής, αλλά και λόγοι όπως:

**Έλεγχος πρόσβασης** -- Αυτός είναι ίσως ο μεγαλύτερος και σπουδαιότερος λόγος για την αποφυγή χρήσης των καθολικών μεταβλητών. Εξ ορισμούς τους κάθε τμήμα κώδικα μπορεί να χρησιμοποιήσει τα δεδομένα τους με όποιον τρόπο θέλει, ανεξάρτητα το λόγο. Οι κανόνες πρόσβασης μπορούν να παρακαμφθούν και τα δεδομένα μπορούν να ακυρωθούν από έναν απρόσεκτα γραμμένο κώδικα. Μη επαληθευμένη πρόσβαση σημαίνει και αναξιόπιστα δεδομένα.

**Συγχρονισμός** -- Ο συγχρονισμός μπορεί να αποτελέσει πραγματικό ζήτημα στις καθολικές μεταβλητές, οι οποίες εξ ορισμού δεν έχουν κανόνες πρόσβασης. Αν δύο στιγμιότυπα, ή συναρτήσεις, προσπελαύνουν ταυτόχρονα τα ίδια στοιχεία, δεν υπάρχει κανένας τρόπος να εγγυηθεί η σειρά με την οποία συμβαίνει η ανάγνωση ή η γραφή δεδομένων.

**Σύνδεση/Κατασκευή/Σαφήνεια.** -- Η χρήση των καθολικών μεταβλητών συνεπάγεται την στενή σύνδεση των συναρτήσεων που τις χρησιμοποιούν. Αυτό από μόνο του δημιουργεί δυσκολία στην κατανόηση της ροής των δεδομένων.

Μερικοί τρόποι αποφυγής των καθολικών μεταβλητών είναι η χρήση:

**Στατικές μεταβλητές/ αντικείμενα** -- Ένας από τους τρόπους να αποφευχθεί η χρήση των καθολικών μεταβλητών είναι η χρήση των στατικών αντικειμένων. Επιτυγχάνεται έτσι, τουλάχιστον το ελάχιστο επίπεδο ασφάλειας, καθώς μπορεί να υπάρχει κάποιος έλεγχος, ή απαγόρευση της πρόσβασης ή και να απομονωθούν, οι καθολικές μεταβλητές ανάλογα με τις ανάγκες της εφαρμογής. Μπορεί αυτό να συνεπάγεται επιπλέον κόπο, αλλά το επίπεδο της ασφάλειας των δεδομένων που μπορεί να επιτευχθεί αξίζει τον κόπο.
**Επαναπροσδιορισμός** -- Πριν τη χρήση των καθολικών μεταβλητών καλό θα ήταν να επαναπροσδιοριστεί ο λόγος για τον οποίο γίνεται αυτή. Πολύ συχνά, για λόγους ευκολίας, χρησιμοποιούνται για την κοινοποίηση δεδομένων ανάμεσα σε συναρτήσεις ή την αποθήκευση πληροφοριών συνόδου (session), ενώ θα μπορούσαν να αντικατασταθούν από αρχεία ρύθμισης (config files) ή δομών συνόδου.

Εντούτοις, υπάρχουν μερικές περιπτώσεις όπου η χρήση των καθολικών μεταβλητών μπορεί να είναι ωφέλιμη, όπως σε πολύ μικρές εφαρμογές όπου η ασφάλεια των δεδομένων δεν παίζει κανένα ρόλο.

***Σημείωση**: Πριν τη χρήση μιας καθολικής μεταβλητής καλό θα ήταν να απαντηθεί η ερώτηση: τι θα γίνει αν τα δεδομένα της μεταβλητή αυτής εκτεθούν; Αν δεν διακινδυνεύεται κάτι σημαντικό από αυτή την έκθεση, τότε μπορεί να χρησιμοποιηθεί καθολική μεταβλητή, διαφορετικά καλό θα ήταν να αποφευχθεί η χρήση της.*

### 5.2.12 Μεταβλητές υπερκαθολικές

Η PHP διαθέτει μερικές πολύ χρήσιμες ενσωματωμένες μεταβλητές καθολικού χαρακτήρα (superglobal). Για αυτές τις μεταβλητές δεν υπάρχει περιορισμός πρόσβασης ή εμβέλειας. Μπορούν να προσπελαστούν από παντού, από οποιοδήποτε αρχείο, οποιαδήποτε συνάρτηση, αντικείμενο, τάξη. Βρίσκονται με τη μορφή πίνακα και περιέχουν διάφορες πληροφορίες. Αυτές οι πληροφορίες μπορεί να αφορούν το λειτουργικό περιβάλλον της PHP ή τις έχει εισάγει ο επισκέπτης σε κάποια φόρμα, κ.α. Οι μεταβλητές αυτές είναι διαθέσιμες στις εκδόσεις της PHP μετά την 4.1. Στην ονοματολογία τους χρησιμοποιούνται μόνο κεφαλαία γράμματα και αρχίζουν πάντα με την κάτω παύλα (εκτός από την `$GLOBALS`).

Ο Πίνακας 5.5 παρακάτω πίνακας παρουσιάζει αυτές τις μεταβλητές.

**Πίνακας 5.5 Υπέρκαθολικές μεταβλητές**

| Μεταβλητή | Περιγραφή |
|---|---|
| $GLOBALS | Περιέχει όλες τις καθολικές μεταβλητές. |
| $_SERVER | Περιέχει πληροφορίες όπως: επικεφαλίδες πρωτοκόλλου, μονοπάτια και τοποθεσίες σεναρίων. Τα περιλαμβανόμενα στοιχεία, και η τιμή, τους παρέχονται από τον εξυπηρετητή Ιστού (web server) και είναι δυνατόν να παρέχονται όλα, κάποια ή κανένα, ανάλογα την εγκατάσταση. |
| $_ENV | Περιέχει μεταβλητές που παρέχονται από το περιβάλλον εγκατάστασης της PHP. |
| $_GET | Περιέχει μεταβλητές που παρέχονται μέσω της μεθόδου HTTP Get. |
| $_POST | Περιέχει μεταβλητές που παρέχονται μέσω της μεθόδου HTTP Post. |
| $_COOKIE | Περιέχει τις μεταβλητές που χρησιμοποιούνται ως HTTP cookies. |
| $_REQUEST | Περιέχει πληροφορίες που παρέχει ο φυλλομετρητής. Αποτελεί γονικό πίνακα των $_GET, $_POST, and $_COOKIE. |
| $_SESSION | Περιέχει τις μεταβλητές που χρησιμοποιούνται σε σύνοδο (session). |
| $_FILES | Περιέχει τα αρχεία που έχουν ‘ανεβεί’ (upload) στον εξυπηρετητή Ιστού με τη χρήση της μεθόδου HTTP Post. |

Πρόκειται για μεταβλητές τύπου συσχετιζόμενου πίνακα, όπου τα κλειδιά των στοιχείων τους αποτελούν τα ονόματα των επιμέρους μεταβλητών που περιέχουν.
Σε όλες, τα ονόματα των μεταβλητών που περιέχονται έχουν οριστεί, με κάποιο τρόπο, από τον κώδικα της εφαρμογής. Για παράδειγμα, έστω ότι το παρακάτω είναι το περιεχόμενο του πίνακα `$_GET`:

```
Array (
            [username]=>anna
            [birthdate]=>1986-01-06
            [gender]=>female
            [age]=>50
```

```
)
```

Αυτό σημαίνει ότι σε κάποιο τμήμα κώδικα της εφαρμογής ορίστηκαν και χρησιμοποιήθηκαν (ή θα χρησιμοποιηθούν) οι μεταβλητές: `$username`, `$birthdate`, `$gender`, `$age`. Οι τιμές των μεταβλητών θα διαφοροποιηθούν κατά τις εκτελέσεις της εφαρμογής.

***Σημείωση**: Η διάθεση των υπερ -καθολικών μεταβλητών ορίζεται από παραμέτρους της εγκατάστασης (php.ini). Η σειρά με την οποία ενεργοποιούνται αυτές οι μεταβλητές έχει επιπτώσεις στην απόδοση της εφαρμογής. Η προκαθορισμένη σειρά είναι: variables_order = "GPCS", όπου GPCS =$_GET, $_POST, $_COOKIE, $_SERVER. Αν δεν οριστεί ρητά, η μεταβλητή $_ENV δεν ενεργοποιείται.*

Εξαίρεση αποτελούν οι πίνακες `$_SERVER` και `$_ENV`, τα στοιχεία των οποίων είναι καθορισμένα από το εξωτερικό περιβάλλον της εφαρμογής. Για παράδειγμα η `$_SERVER` περιέχει πληροφορίες που παρέχονται από την εγκατάσταση του εξυπηρετητή Ιστού. Το παρακάτω σενάριο κώδικα αποτελεί μερικό αποτέλεσμα της εντολής `print_r($_SERVER);`.

```
Array
(
    [DB_USER] => ***
    [DB_PASS] => ***
    [HTTP_HOST] => localhost:8888
    [HTTP_USER_AGENT] => Mozilla/5.0 ...
    [HTTP_ACCEPT] =>
text/html,application/xhtml+xml,application/xml;****
    [HTTP_ACCEPT_LANGUAGE] => en-us,en;q=0.5
    [HTTP_ACCEPT_ENCODING] => gzip, deflate
    [HTTP_CONNECTION] => ....
    [PATH] => ....
……
……
)
```

Ο xxxxxxxxxxΠίνακας 5.6 πίνακας περιέχει τις πιο δημοφιλείς μεταβλητές του πίνακα `$_SERVER`.

**Πίνακας 5.6 Περιεχόμενες σταθερές της μεταβλητής πίνακα $_SERVER**

| Σταθερά | Περιέχει |
|---|---|
| PHP_SELF | Το όνομα του τρέχοντος σεναρίου, σε σχέση με το ριζικό κατάλογο εγγράφων, π.χ. εάν ισχύει το http://example.com/foo/bar.php, τότε θα έχει τιμή την /foo/bar.php. |
| SERVER_ADDR | Η διεύθυνση IP του εξυπηρετητή Ιστού όπου εκτελείται το τρέχον σενάριο κώδικα. |
| SERVER_NAME | Το όνομα του εξυπηρετητή Ιστού (εικονικού ή πραγματικού) όπου εκτελείται το τρέχον σενάριο κώδικα, π.χ. www.unipi.gr. |
| SERVER_SOFTWARE | Την αλφαριθμητική ταυτότητα του εξυπηρετητή Ιστού, π.χ. Apache/2.2.24. |
| SERVER_PROTOCOL | Το πρωτόκολλο μέσω του οποίου παρουσιάστηκε η ιστοσελίδα στον επισκέπτη, π.χ. 'HTTP/1.0'. |
| REQUEST_METHOD | Τη μέθοδο που χρησιμοποιήθηκε στην πρόσβαση της ιστοσελίδας, π.χ. 'GET', 'HEAD', 'POST', 'PUT'. |
| REQUEST_TIME | Τη χρονική σήμανση της έναρξης της αίτησης, π.χ. 1377687496. Ισχύει από την έκδοση PHP 5.1.0. |
| QUERY_STRING | Το αίτημα στη βάση, εάν υπάρχει, που χρησιμοποιήθηκε στην πρόσβαση της ιστοσελίδας. |
| DOCUMENT_ROOT | Το ριζικό κατάλογο του τρέχοντος σεναρίου, όπως αυτός έχει οριστεί στο αρχείο ρυθμίσεων (configuration file). |
| GATEWAY_INTERFACE | Την έκδοση του Common Gateway Interface (CGI) που χρησιμοποιεί ο εξυπηρετητής Ιστού. |

| | |
|---|---|
| HTTP_ACCEPT | Τα στοιχεία της κεφαλίδας ‘accept’ του πρωτοκόλλου HTTP. |
| HTTP_ACCEPT_CHARSET | Τα στοιχεία της κεφαλίδας ‘Accept_Charset’ του πρωτοκόλλου HTTP, π.χ. utf-8, ISO-8859-1 |
| HTTP_CONNECTION | Τα στοιχεία της κεφαλίδας ‘connection’ του πρωτοκόλλου HTTP, του τρέχοντος αιτήματος, π.χ. 'Keep-Alive'. |
| HTTP_HOST | Τα στοιχεία της κεφαλίδας ‘host’ του πρωτοκόλλου HTTP. |
| HTTP_REFERER | Όταν ένας φυλλομετρητής μετακινείται ανάμεσα σε ιστοτόπους ή/και ανάμεσα στις σελίδες ενός δικτυακού τόπου, μπορεί προαιρετικά να παραθέσει τη διεύθυνση URL από όπου ήρθε.<br>Χρησιμοποιείται με επιφύλαξη καθώς δεν υποστηρίζεται από όλους τους φυλλομετρητές. |
| HTTP_USER_AGENT | Περιγράφει το σύστημα πρόσβασης (φυλλομετρητή, λειτουργικό, γλώσσα κλπ) του επισκέπτη, π.χ. Mozilla/4.5 [en] (X11; U; Linux 2.2.9 i586). |
| HTTPS | Εάν χρησιμοποιείται το πρωτόκολλο HTTPS, τότε περιέχει μια μη κενή τιμή. |
| REMOTE_ADDR | Τη διεύθυνση IP του επισκέπτη. |
| REMOTE_HOST | Το όνομα του υπολογιστή του επισκέπτη της ιστοσελίδας. |
| REMOTE_PORT | Τη θύρα επικοινωνίας του μηχανήματος του επισκέπτη με τον εξυπηρετητή Ιστού. |
| SCRIPT_FILENAME | Το πλήρες μονοπάτι του αρχείου του τρέχοντος σεναρίου. |
| SERVER_ADMIN | Την τιμή της σταθεράς SERVER_ADMIN (για τον Apache) που περιέχεται στο αρχείο ρυθμίσεων. |

| | |
|---|---|
| SERVER_PORT | Τη θύρα επικοινωνίας του μηχανήματος που φιλοξενεί τον εξυπηρετητή Ιστού. Η προκαθορισμένη θύρα είναι η 80. |
| SERVER_SIGNATURE | Πληροφορίες για την έκδοση και το όνομα του εικονικού εξυπηρετητή οι οποίες δύνανται να προστεθούν στις παραγόμενες ιστοσελίδες. |
| PATH_TRANSLATED | Το μονοπάτι του τρέχοντος σεναρίου όπως αυτό εμφανίζεται στο σύστημα καταλόγων του εξυπηρετητή. |
| SCRIPT_NAME | Το μονοπάτι του τρέχοντος σεναρίου. |
| REQUEST_URI | Η διεύθυνση URI που δόθηκε κατά την πρόσβαση της ιστοσελίδας, π.χ. '/index.html'. |
| PHP_AUTH_USER | Όταν πραγματοποιείται αυθεντικόποιηση μέσω του HTTP, περιέχει το όνομα του χρήστη. |
| PHP_AUTH_PW | Όταν πραγματοποιείται αυθεντικόποιηση μέσω του HTTP, περιέχει το συνθηματικό του χρήστη. |
| AUTH_TYPE | Όταν πραγματοποιείται αυθεντικόποιηση μέσω του HTTP, περιέχει τον τύπο της αυθεντικόποιησης. |

***Σημείωση**: Όλες οι σταθερές του πίνακα $_SERVER, οι οποίες περιέχουν πληροφορίες από το πρωτόκολλο HTTP, χρησιμοποιούν ονόματα που αρχίζουν με το λεκτικό HTTP_.*

Το παρακάτω σενάριο παρέχει πληροφορίες για τον επισκέπτη της εφαρμογής.

```
<?php
if (isset($_SERVER['REMOTE_ADDR']))
echo "Διεύθυνση IP: " . $_SERVER['REMOTE_ADDR'] . "\n";
if (isset ($_SERVER['REMOTE_HOST']))
echo "Μηχάνημα: " . $_SERVER['REMOTE_HOST'] . "\n";
```

```
if (isset($_SERVER['REMOTE_PORT']))
echo "Θύρα: " . $_SERVER['REMOTE_PORT'] . "\n";
?>
```

Αρχικά ελέγχει εάν έχουν οριστεί οι σταθερές `REMOTE_ADDR, REMOTE_HOST, REMOTE_PORT`, με τη χρήση της συνάρτησης `isset()`. Εάν οι σταθερές έχουν οριστεί, άρα έχουν και τιμές, τότε τις εκτυπώνει. Το αποτέλεσμα θα είναι κάτι όπως το παρακάτω:

```
Διεύθυνση IP: [89.210.84.187]
Μηχάνημα: some_pc.some_net.net
Θύρα: 40188
```

Η `$_ENV` περιέχει πληροφορίες που προέρχονται από το περιβάλλον λειτουργία της PHP, την εγκατάσταση της, τον φλοιό εκτέλεσης κλπ. Εξ αιτίας αυτού, τα ονόματα των μεταβλητών που περιέχει, δύναται να διαφέρουν από εξυπηρετητή σε εξυπηρετητή. Έστω ότι το παρακάτω αποτελεί τμήμα από το περιεχόμενο της `$_ENV`:

```
Array
(
    [ALLUSERSPROFILE] => C:\ProgramData
    [CommonProgramFiles] => C:\Program Files\Common Files
    [COMPUTERNAME] => ***
    [ComSpec] => C:\Windows\system32\cmd.exe
    [HOMEDRIVE] => C:
    [HOMEPATH] => \Users\***
    [LOGONSERVER] => \\***
    [NUMBER_OF_PROCESSORS] => 2
    [OS] => Windows_NT
    [Path] => C:\Windows\system32;C:\Windows;
    [PROCESSOR_ARCHITECTURE] => x86
```

..... .

..... .

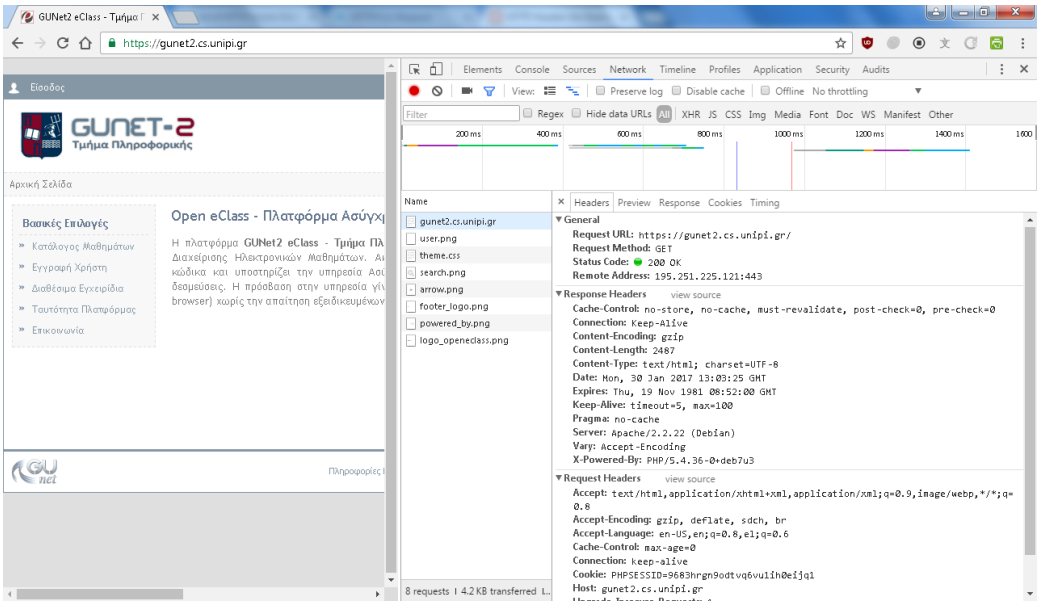
)

Οι πληροφορίες που εμφανίζει περιέχονται σε σταθερές.

***Σημείωση**: Τονίζεται ότι ο πίνακας $_ENV δεν ορίζει μια μεταβλητή περιβάλλοντος, δηλαδή η μεταβλητή δεν θα μεταβιβαστεί σε οποιοδήποτε εξαρτώμενη διαδικασία (child processes). Για να οριστούν πραγματικά μεταβλητές περιβάλλοντος, πρέπει να χρησιμοποιηθεί η συνάρτηση putenv() και η getenv() ώστε να διαβαστούν.*
*Η προσθήκη μιας μεταβλητής στον πίνακα $_ENV σημαίνει μόνο τη δημιουργία μιας υπέρ-καθολικής μεταβλητής, η οποία δεν επηρεάζει το περιβάλλον λειτουργίας της PHP.*
*Δεν εμφανίζεται κανένα προειδοποιητικό μήνυμα όταν γίνεται εγγραφή στον πίνακα $_ENV.*

***Σημείωση**: Αποτελεί καλή πρακτική η αποφυγή της απευθείας πρόσβασης στις μεταβλητές αυτού του είδους. Συνίσταται τα δεδομένα των μεταβλητών αυτών να χρησιμοποιούνται αφού πρώτα έχουν ελεγχθεί από κάποια συνάρτηση φιλτραρίσματος.*

Στην Εικόνα 5.1 μπορεί κανείς να παρατηρήσει τις πληροφορίες που περιέχουν οι επικεφαλίδες του πρωτοκόλλου HTTP. Πολλές από αυτές τις πληροφορίες είναι διαθέσιμες στον κώδικα PHP μέσω των υπερ καθολικών μεταβλητών που μελετήθηκαν σε αυτή την ενότητα. Έτσι ισχύει ότι το περιεχόμενο της $_SERVER[‘HTTP_HOST’] θα είναι gunet2.cs.unipi.gr, ενώ της $_COOKIE[‘PHPSESSID’] θα είναι 9683hrgn9odtvq6vu1ih0eijq1. Η μεταβλητή $_SERVER['REQUEST_METHOD'] φανερώνει ότι η μέθοδος με την οποία έχουν σταλεί τα δεδομένα της φόρμας στον εξυπηρετητή είναι η GET.




**Εικόνα 5.2 – Προβολή επικεφαλίδων HTTP.**

***Σημείωση**: Στο φυλλομετρητή chrome τα περιεχόμενα των επικεφαλίδων του πρωτοκόλλου HTTP μπορούν να αναγνωστούν κάνοντας δεξί κλικ και επιλέγοντας inspect και στη συνέχεια την καρτέλα network. Στην επόμενη ανανέωση της σελίδας εμφανίζονται οι ζητούμενες πληροφορίες.*

### 5.2.13 Μέθοδοι GET και POST

Οι φόρμες αποτελούν το μέσο συλλογής πληροφοριών από τον επισκέπτη μιας εφαρμογής. Ο ορισμός μιας τυπικής φόρμας περιλαμβάνει τη συμπλήρωση δυο χαρακτηριστικών: της μεθόδου (method) με την οποία θα σταλούν τα δεδομένα και των ενεργειών (action) που θα πραγματοποιηθούν σε αυτά.

Πριν την αποστολή των πληροφοριών, αυτές κωδικοποιούνται σύμφωνα με κάποιους κανόνες που ονομάζονται σχήμα URL. Σύμφωνα με αυτό το σχήμα οι μεταβλητές και οι τιμές τους συνδέονται με το σύμβολο της ισότητας (=) σχηματίζοντας ζεύγη, ενώ τα διαφορετικά ζεύγη διαχωρίζονται μεταξύ τους με το σύμβολο &. Τα διαστήματα αντικαθίστανται από το σύμβολο + και οποιοσδήποτε μη αλφαριθμητικός χαρακτήρας αντικαθίσταται από τη δεκαεξαδική του τιμή. Τα διαφορετικά ήδη πληροφοριών διαχωρίζονται από το σύμβολο ?.

Παρακάτω εξετάζονται οι δυο μέθοδοι που μπορούν να χρησιμοποιηθούν για την αποστολή πληροφοριών: η GET και η POST.
Στο επόμενο παράδειγμα παρουσιάζεται μια φόρμα, η οποία χρησιμοποιεί τη μέθοδο GET.

```
<form action="do_something_script.php" method="get">

<input type="text" name="name" placeholder="Όνομα"><br>

<input type="text" name="email" placeholder="Email"><br>

<input type="text" name="contact" placeholder="Κιν.
Τηλέφωνο"><br>

<input type="submit" name="submit" value="Submit">

</form>
```

Όταν ο επισκέπτης επιλέξει το κουμπί "Submit" τότε ο φυλλομετρητής θα κωδικοποιήσει τις πληροφορίες, σύμφωνα με το ανωτέρω σχήμα, ως εξής:

```
http://www.example.com/do_something_script.php?name=john&e
mail=john@gmail.com&contact=9877989898
```

Αυτές οι πληροφορίες θα είναι ορατές στον καθένα στο τμήμα της διεύθυνσης του φυλλομετρητή.
Στην πλευρά του εξυπηρετητή, το αρχείο do_something_script.php μπορεί να περιέχει κάτι σαν το παρακάτω:

```
<?php
switch($_SERVER['REQUEST_METHOD'])
{
case 'GET':
```

```
if( $_GET["name"] || $_GET["email"] || $_GET["contact"])
{
echo "Καλωήρθες: ". $_GET['name']. "<br>";
echo "Το Email σου είναι: ". $_GET["email"]. "<br>";
echo "Το τηλέφωνό σου είναι: ". $_GET["contact"];
}
break;
case 'POST':
if( $_ POST["name"] || $_POST["email"] ||
$_POST["contact"])
{
echo "Καλωήρθες: ". $_POST['name']. "<br />";
echo "Το Email σου είναι: ". $_POST["email"]. "<br />";
echo "Το τηλέφωνό σου είναι: ". $_POST["contact"];
}
break;

?>
```

Ανάλογα με ποια μέθοδο έχει χρησιμοποιηθεί στη φόρμα της εφαρμογής, τότε, στη πλευρά του εξυπηρετητή οι τιμές των μεταβλητών θα αναζητηθούν στην αντίστοιχη υπερ καθολική μεταβλητή ($_GET ή $_POST).

***Σημείωση**: Παρά το γεγονός ότι η μεταβλητή πίνακα $_REQUEST αποτελεί συγχώνευση των μεταβλητών $_POST και $_GET, οπότε περιέχει όλες τις πληροφορίες τους, εντούτοις θα πρέπει να αποφεύγεται η χρήση της, καθώς ελλοχεύουν κίνδυνοι ασφαλείας. Δεν υπάρχει εγγύηση ότι τα δεδομένα της $_REQUEST προέρχονται πραγματικά από την $_POST ή την $_GET.Επιπλέον αν υπάρχει μεταβλητή με το ίδιο όνομα και στην $_POST και την $_GET, τότε αυτή που προέρχεται από την $_POST θα αποδίδεται πρώτη.*

Γενικά η μέθοδος GET διαφέρει σε σχέση με την POST στα παρακάτω:

- ✓ Με τη μέθοδο GET τα δεδομένα της φόρμας εμφανίζονται στο τμήμα της διεύθυνσης του φυλλομετρητή.

- ✓ Η μέθοδος GET δεν ενδείκνυται για την αποστολή προσωπικών πληροφοριών καθώς αυτές είναι ορατές, σύμφωνα με τα προηγούμενα.
- ✓ Όταν χρησιμοποιείται η μέθοδος GET τότε μπορεί να εισαχθεί σελιδοδείκτης (bookmark) στη σελίδα της συμπληρωμένης φόρμας, οπότε και την καθιστά προσβάσιμη στον καθένα.
- ✓ Όταν χρησιμοποιείται η μέθοδος GET τότε η σελίδα της συμπληρωμένης φόρμας παραμένει στο ιστορικό του φυλλομετρητή.
- ✓ Η μέθοδος GET παρουσιάζει περιορισμό στο μέγεθος των δεδομένων που μπορεί να στείλει, καθώς το URL έχει μέγιστο μήκος 2048 χαρακτήρες.
- ✓ Η μέθοδος GET εξαιτίας του περιορισμού μεγέθους δεν μπορεί να χρησιμοποιηθεί ώστε να στείλει όλους τους τύπους δεδομένων, π.χ. binary. Χρησιμοποιείται για την αποστολή ASCII δεδομένων.

### 5.2.14 Φόρμες

Ο σκοπός μιας φόρμας αποτελεί την συλλογή και αποστολή πληροφοριών στον εξυπηρετητή Ιστού. Στην πλευρά του πελάτη πραγματοποιείται ο σχεδιασμός /παρουσίασή της, με τη χρήση των ετικετών HTML. Στην πλευρά του εξυπηρετητή πραγματοποιείται η παραλαβή και επεξεργασία των πληροφοριών αυτών. Στη συνέχεια είναι δυνατό να σταλεί κάποια απάντηση στον πελάτη.
Στη συνέχεια της ενότητας αυτής θα εξεταστούν η δημιουργία και επεξεργασία διαφόρων φορμών, οι οποίες παρουσιάζουν ενδιαφέρουσες λειτουργίες.

http://www.w3schools.com/php/php_forms.asp
http://www.html-form-guide.com/php-form/php-form-tutorial.html
http://www.html-form-guide.com/php-form/php-form-validation-tutorial.html

https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Forms/Sending_and_retrieving_form_data

http://www.html-form-guide.com/category/php-form

**Περισυλλογή δεδομένων**

Στο παρακάτω παράδειγμα χρησιμοποιείται μια ιστοσελίδα που περιέχει φόρμα με πεδία για τη συλλογή του ονόματος και των στοιχείων μιας διεύθυνσης, τα οποία, στη συνέχεια, παρουσιάζονται σε άλλη ιστοσελίδα.

Η πρώτη ιστοσελίδα που περιέχει τη φόρμα συλλογής στοιχείων, χρησιμοποιώντας απλά στοιχεία HTML, ονομάζεται "myform.html".

```
<!DOCTYPE html>
<html>
 <head>
     <title>Πρώτη ιστοσελίδα με φόρμα</title>
      <meta charset="UTF-8">
 </head>
 <body>

  <form action="address.php" method="post">

    <label for="name">Το όνομά μου είναι:</label>
    <br> <input type="text" id="name" name="YourName" >
      <br><br>
      <label for="mail">Το email μου είναι:<label> <br>
      <input type="email" id="mail" name="YourEmail">
      <br><br>
      <label for="addr">Η διεύθυνσή μου είναι:<label> <br>
      <input type="text" id="addr" name="YourAddr">
      <br><br>
      <input type="submit" name="submit" value="Send">

  </form>
 </body>
</html>
```

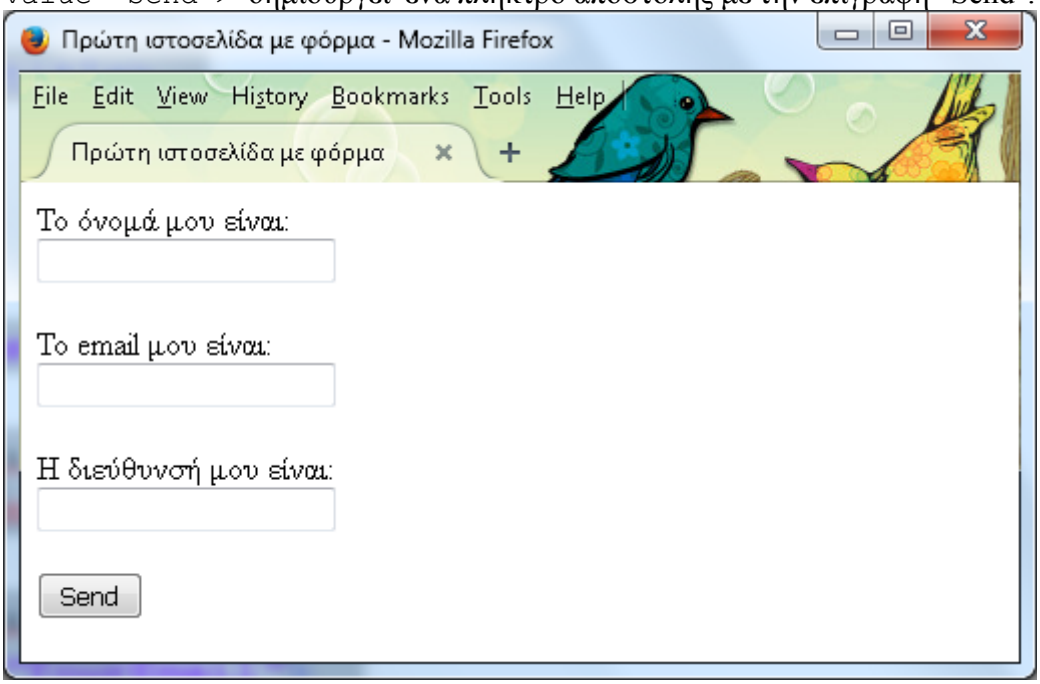
Αυτή είναι μία απλή φόρμα σε HTML, όπως φαίνεται και στην Εικόνα 5.3.1. Αναλυτικότερα ο παραπάνω κώδικας με τη δήλωση `<form action="address.php" method="post">` ορίζει στο φυλλομετρητή ποιο έγγραφο PHP θα επεξεργάζεται τα αποτελέσματα της φόρμας, καθώς και ποια μέθοδος (Post ή Get) θα χρησιμοποιηθεί για την αποστολή των δεδομένων στον εξυπηρετητή.

Στη συνέχεια με τη δήλωση `<input type="text" name="YourName">` καθορίζεται το στοιχείο της φόρμας που θα συλλέγει το όνομα του χρήστη. Είναι ένα πεδίο κειμένου. Η δήλωση `name="YourName"` καθορίζει πως οτιδήποτε γράψει ο χρήστης μέσα στο πεδίο κειμένου θα αποθηκευτεί σε μία μεταβλητή που καλείται "`YourName`".

Το πεδίο που χρησιμοποιείται για την εισαγωγή της διεύθυνσης ηλεκτρονικού ταχυδρομείου είναι το `<input type="email" id="mail" name="YourEmail">`.

Για τη συλλογή της διεύθυνσης του χρήστη χρησιμοποιείται το στοιχείο: `<input type="text" name="YourAddr">`.

Η δήλωση `<input type="submit" name="submit" value="Send">` δημιουργεί ένα πλήκτρο αποστολής με την επιγραφή "Send".



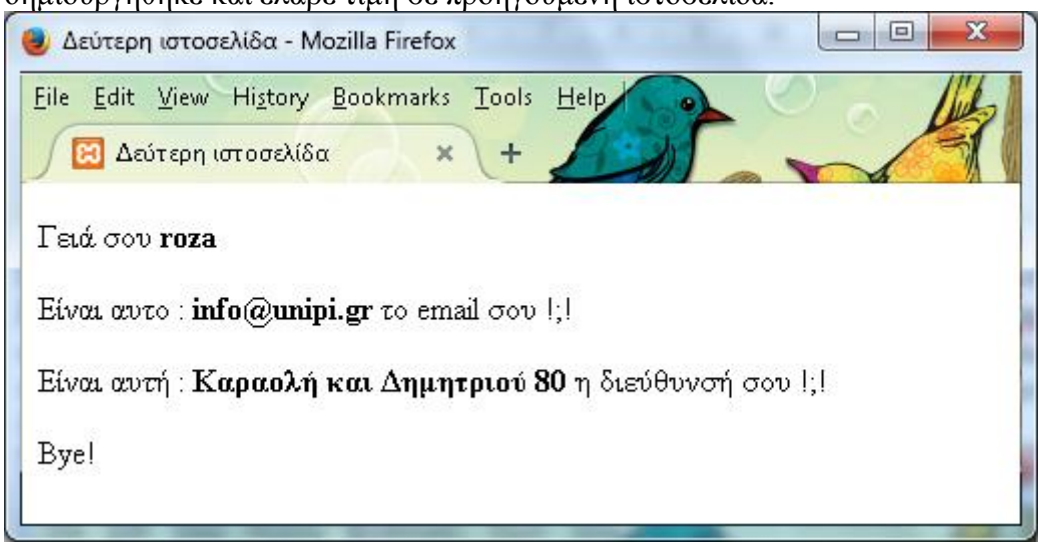

**Εικόνα 5.3 – Φόρμα εισαγωγής δεδομένων**

Παρακάτω θα δούμε την δομή του και το περιεχόμενο του αρχείου `address.php` το οποίο επεξεργάζεται τα δεδομένα εισόδου της προηγούμενης φόρμας. Το αρχείο αυτό έχει τον ακόλουθο κώδικα και το αποτέλεσμα του εμφανίζεται στην Εικόνα 5.3.2:

```
<!DOCTYPE html>
<html>
<head>
```

```
<title>Δεύτερη ιστοσελίδα</title>
<meta charset="UTF-8">
</head>

<body>

<p>Γειά σου <strong>
<?php echo $_POST['YourName'];?>
</strong></p>
<p>Είναι αυτο : <strong>
<?php echo $_POST['YourEmail']; ?>
</strong> το email σου !;! </p>
<p>Είναι αυτή : <strong>
<?php echo $_POST['YourAddr']; ?>
</strong> η διεύθυνσή σου !;! </p>
<p>Bye!</p>
</body>
</html>
```

Για την αποστολή των δεδομένων της φόρμας χρησιμοποιήθηκε η μέθοδος post. Εξ' αιτίας αυτού οι μεταβλητές που περιείχε η φόρμα αναζητήθηκαν στον υπερ καθολικό πίνακα $_POST. Έτσι, στο παραπάνω παράδειγμα η δήλωση `echo $_POST['YourName']` εκτυπώνει την μεταβλητή `YourName` που δημιουργήθηκε και έλαβε τιμή σε προηγούμενη ιστοσελίδα.




**Εικόνα 5.4 – Αποτελέσματα εκτέλεσης σεναρίου PHP**

**Επικύρωση πεδίων**

Μέσω των φορμών οι επισκέπτες μιας ιστοσελίδας, προσκαλούνται να εισάγουν/ στείλουν πληροφορίες. Δυστυχώς δεν συμπληρώνουν όλοι σωστά τα απαιτούμενα πεδία. Έστω, για παράδειγμα, ότι ζητείται να συμπληρωθεί ένας τηλεφωνικός αριθμός. Μερικοί θα σημειώσουν δέκα ψηφία (π.χ. 2104142000), άλλοι δεκατέσσερα καθώς θα περιλάβουν και τον διεθνή κωδικό κλήσης (00302104142000), ενώ κάποιοι άλλοι θα περιλάβουν και σύμβολα (π.χ.+302104142000). Επιπλέον υπάρχουν και οι κακόβουλοι χρήστες που δύναται να επιχειρήσουν να 'αλλάξουν' την ιστοσελίδα, είτε να αποκτήσουν πρόσβαση σε 'ιδιωτικές' περιοχές της.

***Σημείωση**: Κατά τη σχεδίαση εφαρμογών διαδικτύου ακολουθείται η αρχή ότι **ποτέ** δεν εμπιστευόμαστε τα δεδομένα που εισάγει ο χρήστης. Διότι μπορεί να μην είναι πάντα κακόβουλος χρήστης αλλά μπορεί πάντα να κάνει λάθος.*

Ανεξαρτήτως της αιτίας, λάθος ή σκόπιμα, οι φόρμες είναι πάντα δυνατόν να συμπληρωθούν με λανθασμένα δεδομένα. Για την αποτροπή εισαγωγής, καθώς και την αναγνώριση, λανθασμένων πληροφοριών ακολουθούνται ορισμένοι κανόνες όπως οι παρακάτω:

- πρέπει να παρέχονται αρκετές οδηγίες και υποδείξεις
- πρέπει να υπάρχει μια λογική σειρά στην πλοήγηση ανάμεσα στα πεδία της φόρμας
- οι χρήστες μαθαίνουν τα λάθη τους κατά την πληκτρολόγηση
- οι φόρμες θα πρέπει να μπορούν να συμπληρωθούν και να αποσταλούν με τη χρήση μόνο του πληκτρολογίου, π.χ. στις συσκευές κινητής τηλεφωνίας.

Ο έλεγχος της εγκυρότητας (input validation) των δεδομένων αποτελεί ευθύνη του προγραμματιστή και πρέπει να πραγματοποιείται πριν τη χρήση τους. Ανάλογα με τις τεχνικές που εφαρμόζονται, ο έλεγχος μπορεί να γίνει: στην πλευρά του πελάτη, του εξυπηρετητή ή και στις δυο.

Στην πρώτη περίπτωση, αξιοποιώντας τα πλεονεκτήματα μερικών νέων χαρακτηριστικών που εισήγαγε η HTML5 ή/και τις προγραμματιστικές δυνατότητες της javascript, μπορεί να ελεγχθεί η εγκυρότητα των δεδομένων στο φυλομετρητή του επισκέπτη, πριν αυτά αποσταλούν στον απομακρυσμένο εξυπηρετητή. Το κυριότερο πλεονέκτημα της περίπτωσης αυτής αποτελεί η ταχύτητα ειδοποίησης του χρήστη. Η λάθος μορφή δεδομένων μπορεί να γίνει αντιληπτή σχεδόν κατά την πληκτρολόγηση και πριν την υποβολή της φόρμας. Το κυριότερο μειονέκτημα της αποτελεί το ενδεχόμενο ο χρήστης να έχει απενεργοποιήσει τεχνολογίες που χρησιμοποιούνται στον έλεγχο, π.χ. την

javascript. Επιπλέον ο φυλλομετρητής μπορεί να μην υποστηρίζει χρησιμοποιούμενα χαρακτηριστικά των πεδίων της φόρμας π.χ. patern, type=mail ή διάφορους κανόνες CSS, π.χ. :valid, :active κλπ.
Στη δεύτερη περίπτωση, όταν ο έλεγχος εφαρμόζεται στη πλευρά του εξυπηρετητή, μπορεί κατά την παραλαβή, και πριν τη χρήση, να ελεγχθεί η εγκυρότητα των δεδομένων, ανάλογα τη γλώσσα που θα εφαρμοστεί (π.χ. php, perl, κ.α.). Το κυριότερο πλεονέκτημα της περίπτωσης αυτής αποτελεί το γεγονός ότι το περιβάλλον που πραγματοποιείται ο έλεγχος είναι σταθερό και καθορίζεται από τον προγραμματιστή. Οι τιμές των πεδίων ελέγχονται μαζικά και μετά την υποβολή της φόρμας.
Εναλλακτικά μπορεί να εφαρμοστεί μερικώς έλεγχος (είτε στη μορφή, είτε σε ορισμένα πεδία) στην πλευρά του πελάτη, κατά την πληκτρολόγηση των δεδομένων. Στη συνέχεια, κατά την παραλαβή των δεδομένων, στην πλευρά του εξυπηρετητή, μπορεί να ολοκληρωθεί ο έλεγχος τους. Ο τελευταίος αυτός τρόπος συνδυάζει τεχνικές από διάφορες τεχνολογίες, π.χ. HTML5, CSS3, javascript, php, perl,κλπ. Αυτή η περίπτωση συνδυάζει τα πλεονεκτήματα των δυο ανωτέρω.
Στα παραδείγματα που ακολουθούν θα εξεταστεί η πραγματοποίηση ελέγχου της φόρμας χρησιμοποιώντας τεχνολογίες τόσο στη πλευρά του πελάτη (π.χ. HTML, CSS) όσο και του εξυπηρετητή (π.χ. PHP).

***Στην πλευρά του πελάτη***

Στο παράδειγμα της ΕΙΚΟΝΑΣΧΧΧ παρουσιάζεται μια φόρμα επικοινωνίας, η οποία ζητά τη συμπλήρωση των: όνομα, email, ιστοσελίδα και κείμενο μηνύματος. Για την επικύρωση των τιμών των πεδίων έχουν χρησιμοποιηθεί τεχνολογίες από την πλευρά του πελάτη, και συγκεκριμένα χαρακτηριστικά της HTML5 καθώς και κανόνες μορφοποίησης της CSS3. Ο κώδικας του παραδείγματος είναι ο παρακάτω.

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8">
 <title>Φόρμα Επικοινωνίας </title>
 <link rel="stylesheet" media="screen" href="styles.css" >
</head>
<body>
```

```
<form class="contact_form"
action="validate_on_server_side.php" method="post"
name="contact_form">
  <ul>
     <li>
       <h2>Επικοινωνία</h2>
       <span class="required_notification">* δηλώνει
απαραίτητο πεδίο</span>
     </li>
     <li>
       <label for="surname">Ονοματεπώνυμο:</label>
       <input type="text" id="surname" name="surname"
placeholder="John Doe" required>
      </li>
      <li>
        <label for="email">Email:</label>
        <input type="email" id="email" name="email"
placeholder="john_doe@unipi.gr" required>
       <span class="form_hint">Αποδεκτή μορφή
"name@something.com"</span>
       </li>
       <li>
         <label for="website">Ιστοσελίδα:</label>
      <input type="url" id="website" name="website"
placeholder="http://johndoe.com" required
pattern="(http|https)://.+">
        <span class="form_hint">Αποδεκτή μορφή
"http://someaddress.com"</span>
        </li>
        <li>
            <label for="message">Μήνυμα:</label>
            <textarea id="message" name="message"
cols="40" rows="6" required ></textarea>
```

```
        </li>
        <li>
      <button class="submit" type="submit">Υποβολή</button>
        </li>
    </ul>
</form>
</body>
</html>
```

Οι κανόνες μορφοποίησης που εφαρμόζονται στη φόρμα εμφανίζονται παρακάτω (αρχείο styles.css).

```
/* === Απομάκρυνση προκαθορισμένων μορφοποιήσεων  === */
*:focus {outline: none;}


/* === Γραμματοσειρά === */
body {font: 14px/21px "Lucida Sans", "Lucida Grande", "Lucida Sans
Unicode", sans-serif;}
.contact_form h2, .contact_form label {font-family:Georgia, Times,
"Times New Roman", serif;}
.form_hint, .required_notification {font-size: 11px;}


/* === Μορφοποίηση λίστας === */
.contact_form ul {
    width:750px;
    list-style-type:none;
       list-style-position:outside;
       margin:0px;
       padding:0px;
}
.contact_form li{
       padding:12px;
       border-bottom:1px solid #eee;
       position:relative;
}
```

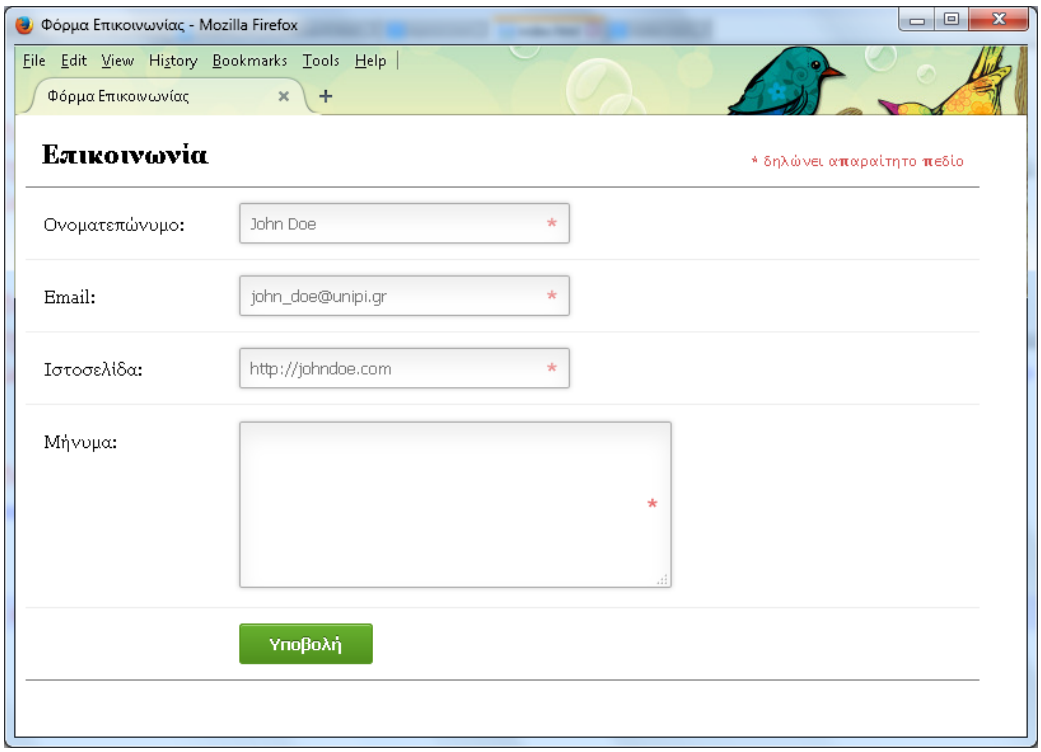
```
.contact_form li:first-child, .contact_form li:last-child {
	border-bottom:1px solid #777;
}

/* === Επικεφαλίδες === */
.contact_form h2 {
	margin:0;
	display: inline;
}
.required_notification {
	color:#d45252;
	margin:5px 0 0 0;
	display:inline;
	float:right;
}

/* === Στοιχεία φόρμας === */
.contact_form label {
	width:150px;
	margin-top: 3px;
	display:inline-block;
	float:left;
	padding:3px;
}
.contact_form input {
	height:20px;
	width:220px;
	padding:5px 8px;
}
.contact_form textarea {padding:8px; width:300px;}
.contact_form button {margin-left:156px;}

	/*  Ορατά στοιχεία φόρμας  */
	.contact_form input, .contact_form textarea {
```

```
            border:1px solid #aaa;
            box-shadow: 0px 0px 3px #ccc, 0 10px 15px #eee inset;
            border-radius:2px;
            padding-right:30px;
            -moz-transition: padding .25s;
            -webkit-transition: padding .25s;
            -o-transition: padding .25s;
            transition: padding .25s;
      }
      .contact_form input:focus, .contact_form textarea:focus {
            background: #fff;
            border:1px solid #555;
            box-shadow: 0 0 3px #aaa;
            padding-right:70px;
      }

/* === HTML5 μορφοποίηση κατά την επικυρωση === */
.contact_form input:required, .contact_form
textarea:required {
     background: #fff url(images/red_asterisk.png) no-
repeat 98% center;
}
.contact_form input:required:valid, .contact_form
textarea:required:valid {
     background: #fff url(images/valid.png) no-repeat 98%
center;
     box-shadow: 0 0 5px #5cd053;
     border-color: #28921f;
}
.contact_form input:focus:invalid, .contact_form
textarea:focus:invalid {
     background: #fff url(images/invalid.png) no-repeat
98% center;
     box-shadow: 0 0 5px #d45252;
```

```
	border-color: #b03535
}

/* === Υποδείξεις φόρμας  === */
.form_hint {
	background: #d45252;
	border-radius: 3px 3px 3px 3px;
	color: white;
	margin-left:8px;
	padding: 1px 6px;
	z-index: 999; /* οι υποδείξεις εμφανίζονται πάνω από τα άλλα στοιχεία */
	position: absolute; /* προσαρμογή αν η υπόδειξη περιλαμβάνει δυο γραμμές */
	display: none;
}
.form_hint::before {
	content: "\25C0";
	color:#d45252;
	position: absolute;
	top:1px;
	left:-6px;
}
.contact_form input:focus + .form_hint {display: inline;}
.contact_form input:required:valid + .form_hint {background: #28921f;}
.contact_form input:required:valid + .form_hint::before {color:#28921f;}

/* === Μορφοποίηση κουμπιού === */
button.submit {
	background-color: #68b12f;
```

```
	background: -webkit-gradient(linear, left top, left bottom,
from(#68b12f), to(#50911e));
	background: -webkit-linear-gradient(top, #68b12f, #50911e);
	background: -moz-linear-gradient(top, #68b12f, #50911e);
	background: -ms-linear-gradient(top, #68b12f, #50911e);
	background: -o-linear-gradient(top, #68b12f, #50911e);
	background: linear-gradient(top, #68b12f, #50911e);
	border: 1px solid #509111;
	border-bottom: 1px solid #5b992b;
	border-radius: 3px;
	-webkit-border-radius: 3px;
	-moz-border-radius: 3px;
	-ms-border-radius: 3px;
	-o-border-radius: 3px;
	box-shadow: inset 0 1px 0 0 #9fd574;
	-webkit-box-shadow: 0 1px 0 0 #9fd574 inset ;
	-moz-box-shadow: 0 1px 0 0 #9fd574 inset;
	-ms-box-shadow: 0 1px 0 0 #9fd574 inset;
	-o-box-shadow: 0 1px 0 0 #9fd574 inset;
	color: white;
	font-weight: bold;
	padding: 6px 20px;
	text-align: center;
	text-shadow: 0 -1px 0 #396715;
}
button.submit:hover {
	opacity:.85;
	cursor: pointer;
}
button.submit:active {
	border: 1px solid #20911e;
	box-shadow: 0 0 10px 5px #356b0b inset;
	-webkit-box-shadow:0 0 10px 5px #356b0b inset ;
	-moz-box-shadow: 0 0 10px 5px #356b0b inset;
```

```
        -ms-box-shadow: 0 0 10px 5px #356b0b inset;
        -o-box-shadow: 0 0 10px 5px #356b0b inset;


}
```




**Εικόνα 5.5 – Επικύρωση φόρμας, χρήση τεχνολογιών CSS και HTML5**

Στο συγκεκριμένο παράδειγμα η επικύρωση των πεδίων επιτυγχάνεται με τις παρακάτω τεχνικές.

- ✓ Έχουν χρησιμοποιηθεί οι ετικέτες label ώστε τα πεδία να είναι σωστά οργανωμένα.
- ✓ Έχει χρησιμοποιηθεί το χαρακτηριστικό required για τα πεδία, τα οποία πρέπει απαραίτητα να συμπληρωθούν. Με τη χρήση αυτού του χαρακτηριστικού, ανάλογα τον φυλλομετρητή, εμφανίζεται προειδοποιητικό μήνυμα, κατά την αιώρηση πάνω από το πεδίο.
- ✓ Έχει χρησιμοποιηθεί το χαρακτηριστικό placeholder, το οποίο παρουσιάζει τιμές για το πεδίο, αποδεκτής μορφής.
- ✓ Έχει χρησιμοποιηθεί το χαρακτηριστικό type="email" ώστε το πεδίο να μπορεί να δεχτεί μόνο διευθύνσεις ηλεκτρονικού ταχυδρομείου.

- ✓ Στο πεδίο της ιστοσελίδας, έχουν χρησιμοποιηθεί τα χαρακτηριστικά: type="url" και pattern= "(http|https)://.+" ώστε να αποτρέπεται η εισαγωγή τιμών που δεν αρχίζουν με http:// ή https://.
- ✓ Έχουν δημιουργηθεί και κατάλληλα μορφοποιηθεί περιοχές στις οποίες εμφανίζονται υποδείξεις συμπλήρωσης, π.χ. .
- ✓ Στους κανόνες μορφοποίησης έχει γίνει πρόβλεψη για κατάλληλη τροποποίηση μορφής στα πεδία που έχουν χαρακτηριστεί ως :valid :invalid.

***Στην πλευρά του εξυπηρετητή***

Στη συνέχεια θα εξεταστεί πώς πραγματοποιείται η επικύρωση των τιμών των πεδίων στη πλευρά του εξυπηρετητή χρησιμοποιώντας τη γλώσσα PHP.

Αρχικά θα πρέπει να βεβαιωθεί ότι έχουν συμπληρωθεί όλα τα απαραίτητα πεδία της φόρμας. για το σκοπό αυτό χρησιμοποιείται η συνάρτηση empty(). Η συνάρτηση αυτή ελέγχει εάν μια μεταβλητή είναι 'κενή', δηλαδή εάν δεν έχει οριστεί, εάν είναι null ή false ή 0 ή περιέχει κενό αλφαριθμητικό (empty string). Έστω ότι πρέπει να ελεγχθεί εάν το πεδίο του ονοματεπώνυμου είναι κενό, στο παράδειγμα της εικόνας XXX. Αυτό πραγματοποιείται εφαρμόζοντας το:

```
if (empty($_POST["surname"])) {
echo "Το Ονοματεπώνυμο δε μπορεί να είναι κενό"; }
```

Στη συνέχεια, έστω ότι χρειάζεται να ελεγχθεί αν το πεδίο του email περιέχει μια διεύθυνση ηλεκτρονικού ταχυδρομείου με έγκυρη μορφή, δηλαδή περιέχει γράμματα, το σύμβολο @ καθώς και μια τελεία (.). Αυτό πραγματοποιείται με τη συνάρτηση filter_var() και εφαρμογή του κατάλληλου φίλτρου, όπως το FILTER_VALIDATE_EMAIL. Για παράδειγμα το παρακάτω:

```
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
  echo "Μη αποδεκτή μορφή email";}
```

Η συνάρτηση filter_var ελέγχει εάν το περιεχόμενο του πρώτου ορίσματος της (εδώ η μεταβλητή $email), είναι σύμφωνο με το σύνολο των κανόνων που ορίζει η δεύτερη παράμετρος της, δηλαδή το φίλτρο της (εδώ το FILTER_VALIDATE_EMAIL). Κάποια άλλα φίλτρα που θα μπορούσαν να χρησιμοποιηθούν σε αυτή τη συνάρτηση παρουσιάζονται στον πίνακα XXX.

**Πίνακας 5.7 Φίλτρα επικύρωσης της PHP.**

| **Φίλτρο** | **Περιγραφή** |
|---|---|
| FILTER_VALIDATE_BOOLEAN | Ελέγχει εάν είναι boolean. |

| | |
|---|---|
| FILTER_VALIDATE_FLOAT | Ελέγχει εάν είναι float. |
| FILTER_VALIDATE_INT | Ελέγχει εάν είναι integer. |
| FILTER_VALIDATE_IP | Ελέγχει εάν είναι διεύθυνση IP. |
| FILTER_VALIDATE_MAC | Ελέγχει εάν είναι διεύθυνση MAC. |
| FILTER_VALIDATE_REGEXP | Ελέγχει εάν ικανοποιείται μια κανονικοποιημένη έκφραση, η οποία είναι συμβατή με η γλώσσα Perl. |
| FILTER_VALIDATE_URL | Ελέγχει εάν είναι URL. Δεν ελέγχει εάν το URL περιλαμβάνει κάποιο έγκυρο πρωτόκολλο, πχ. http, https, ssh κλπ. |

Εφαρμόζοντας το φίλτρο FILTER_VALIDATE_URL στη συνάρτηση filter_var, μπορεί να γίνει έλεγχος εάν μια τιμή αποτελεί έγκυρη μορφή URL. Εντούτοις θα πρέπει να εφαρμοστεί επιπλέον έλεγχος εάν το πρωτόκολλο που, ενδεχομένως, περιλαμβάνεται είναι έγκυρο. Για παράδειγμα η επικύρωση: filter_var('a://google.com', FILTER_VALIDATE_URL), θα επιστρέψει την τιμή 'αληθής'.

Ένας διαφορετικός τρόπος για έλεγχο της τιμής URL γίνεται με την εφαρμογή της κατάλληλης κανονικοποιημένης έκφρασης (regular expression) στη συνάρτηση preg_match. Η συνάρτηση αυτή επιβεβαιώνει την ύπαρξη (ή όχι) ενός δοσμένου μοτίβου (pattern) μέσα σε ένα αλφαριθμητικό. Το μοτίβο παρουσιάζεται με όρους κανονικοποιημένων εκφράσεων. Επιστρέφει την τιμή 1 εφόσον βρεθεί ταύτιση, διαφορετικά 0 και 'false' εφόσον προκύψει κάποιο λάθος.

Οι κανονικοποιημένες εκφράσεις (regular expressions) αποτελούν ακολουθίες από σύμβολα και χαρακτήρες που εκφράζουν ένα αλφαριθμητικό ή μοτίβο και πρέπει να αναζητηθούν μέσα σε ένα μεγαλύτερο κομμάτι του κειμένου. Για παράδειγμα η έκφραση

- ✓ /abc/ σημαίνει ταύτιση της ακολουθίας abc, οπουδήποτε και αν βρίσκεται
- ✓ /abc/i σημαίνει ταύτιση της ακολουθίας abc, οπουδήποτε και αν βρίσκεται, αλλά χωρίς ταύτιση πεζών κεφαλαίων γραμμάτων
- ✓ /^abc/ σημαίνει ταύτιση της ακολουθίας abc, εφόσον αυτή βρίσκεται στην αρχή του αλφαριθμητικού
- ✓ /abc$/ σημαίνει ταύτιση της ακολουθίας abc, εφόσον αυτή βρίσκεται στο τέλος του αλφαριθμητικού
- ✓ /ab./ σημαίνει ταύτιση της ακολουθίας ab ακολουθούμενη από οποιονδήποτε χαρακτήρα(ες)

- ✓ /abc*/ σημαίνει ταύτιση της ακολουθίας ab ακολουθούμενη 0 ή περισσότερους χαρακτήρες c
- ✓ /abc?/ σημαίνει ταύτιση της ακολουθίας ab ακολουθούμενη 0 ή 1 χαρακτήρες c
- ✓ [xyz] σημαίνει ταύτιση με οποιονδήποτε χαρακτήρα περιέχεται στις παρενθέσεις []
- ✓ [a-z] σημαίνει ταύτιση με οποιονδήποτε χαρακτήρα της σειράς που περιέχεται στις παρενθέσεις
- ✓ [^xyz] σημαίνει ταύτιση με οποιονδήποτε χαρακτήρα δεν περιέχεται στις παρενθέσεις []
- ✓ x|y σημαίνει ταύτιση με τον χαρακτήρα x ή τον y.

Το παρακάτω τμήμα κώδικα ελέγχει εάν η μεταβλητή $website περιέχει κάποιο URL έγκυρης μορφής.

```
if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-
9+&@#\/%?=~_|!:,.;]*[-a-z0-9+&@#\/%=~_|]/i",$website)) {
        echo "Μη αποδεκτή μορφή URL";}
```

Στον πίνακα XXX παρουσιάζεται η επεξήγηση της κανονικοποιημένης έκφρασης του προηγούμενου παραδείγματος.

| Έκφραση | Περιγραφή |
|---|---|
| \b | Αναζητά ταύτιση στην αρχή ή το τέλος λέξης (εδώ στην αρχή) |
| (?:https?\|ftp) :\/\/ | Πρώτη εναλλακτική |
| (?:https?\|ftp) | Πρώτη ομάδα ταύτισης. Αναζητά κάποιο από τα: http(s) ή ftp |
| :\/\/ | Αναζητά ταύτιση για :// |
| www\. | Δεύτερη εναλλακτική. Ψάχνει για www. |
| (?:(?:https?\|ftp):\/\/\|www\.) | Δεύτερη ομάδα ταύτισης. Συνδυάζοντας τα προηγούμενα είναι: ((http(s) ή ftp):// ) ή www.) |
| [-a-z0-9+&@#\/%?=~_\|!:,.;] | Αναζητά ταύτιση για κάποιο από τα παρακάτω. |
| - | Αναζητά ταύτιση για τον χαρακτήρα - |
| a-z | Αναζητά ταύτιση για κάποιο χαρακτήρα από το σύνολο a,b,c…z |
| 0-9 | Αναζητά ταύτιση για κάποιο χαρακτήρα από το διάστημα 0….9 |

| | |
|---|---|
| +&@# | Αναζητά ταύτιση για κάποιο χαρακτήρα από τη λίστα +&@# |
| \/ | Αναζητά ταύτιση για το σύμβολο / |
| %?=~_\|!:,.; | Αναζητά ταύτιση για κάποιο χαρακτήρα από τη λίστα %?=~_\|!:,.; |
| * | Αναζητά ταύτιση για 0 ή περισσότερες επαναλήψεις κάποιων από τους χαρακτήρες που περιλήφθηκαν στην παρένθεση |
| [-a-z0-9+&@#\/%=~_\|] | Αναζητά ταύτιση για κάποιο από τα παρακάτω. |
| - | Αναζητά ταύτιση για τον χαρακτήρα - |
| a-z | Αναζητά ταύτιση για κάποιο χαρακτήρα από το σύνολο a,b,c…z |
| 0-9 | Αναζητά ταύτιση για κάποιο χαρακτήρα από το διάστημα 0….9 |
| +&@# | Αναζητά ταύτιση για κάποιο χαρακτήρα από τη λίστα +&@# |
| \/ | Αναζητά ταύτιση για το σύμβολο / |
| %=~_\| | Αναζητά ταύτιση για κάποιο χαρακτήρα από τη λίστα %=~_\| |
| i | Γενική Παράμετρος, η οποία ορίζει ότι η αναζήτηση θα πραγματοποιηθεί χωρίς διάκριση πεζών κεφαλαίων. |

***Σημείωση***: *Στο διαδίκτυο κυκλοφορούν διάφορες εφαρμογές που επικυρώνουν και επεξηγούν κανονικοποιημένες εκφράσεις. Μια καλή επιλογή αποτελεί και η https://regex101.com. Ενώ στο http://regexlib.com/ μπορεί κανείς να αναζητήσει κανονικοποιημένες εκφράσεις που καλύπτουν ένα ευρύ φάσμα περιπτώσεων.*

Εφαρμόζοντας τα ανωτέρω παραδείγματα στη φόρμα της εικόνας ΕΙΚΟΝΑΣΧΧΧ (προηγούμενη εικόνα) δημιουργείται ο παρακάτω κώδικας.

```
<?php
// αρχικοποίηση μεταβλητών με μηδενικές τιμές
$nameErr = $emailErr = $genderErr = $websiteErr = "";
```

```
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["surname"])) {
   $nameErr = "Το Ονοματεπώνυμο δε μπορεί να είναι κενό.";
  } else {
    $name = sanit_input($_POST["surname"]);
    //Έλεγχος ότι περιέχει μόνο γράμματα και κενά
διαστήματα
    if (!preg_match("/^[a-zA-Z ]*$/",$name)) {
      $nameErr = "Στο Ονοματεπώνυμο επιτρέπονται μόνο
γράμματα και κενά διαστήματα.";
    }
  }
  if (empty($_POST["email"])) {
    $emailErr = "Το Email δε μπορεί να είναι κενό.";
  } else {
    $email = sanit_input($_POST["email"]);
    // Έλεγχος έγκυρης μορφής email
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
      $emailErr = "Μη αποδεκτή μορφή email.";
    }
  }

  if (empty($_POST["website"])) {
    $website = "";
  } else {
    $website = sanit_input($_POST["website"]);
    // Έλεγχος έγκυρης μορφής URL)
```

```
    if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-
z0-9+&@#\/%?=~_|!:,.;]*[-a-z0-9+&@#\/%=~_|]/i",$website))
{
      $websiteErr = "Μη αποδεκτή μορφή URL";
    }
  }
  if (empty($_POST["comment"])) {
    $comment = "";
  } else {
    $comment = sanit_input($_POST["comment"]);
  }
}
function sanit_input($data) {
/*Πριν τη χρήση των δεδομένων επιβεβαίωσε ότι δεν
περιέχουν κακόβουλο λογισμικό, τμήμα κώδικα ή λάθος
χαρακτήρα ειδικού σκοπού π.χ. ; */
  return $data;
}
if (!empty($nameErr)) echo $nameErr;
if (!empty($emailErr)) echo $emailErr;
if (!empty($websiteErr)) echo $websiteErr;
?>
```

Οι πληροφορίες που έχει εισάγει ο χρήστης πριν χρησιμοποιηθούν, δηλαδή πριν αποθηκευτούν σε κάποια βάση δεδομένων, σε κάποιο αρχείο ή απλά πριν εκτυπωθούν στην οθόνη του φυλλομετρητη, θα πρέπει να ‘απολυμανθούν’ (sanitize). Αυτό σημαίνει ότι πρέπει να επιβεβαιωθεί ότι δεν περιέχουν κακόβουλο λογισμικό, τμήμα κώδικα ή λάθος χαρακτήρα ειδικού σκοπού. Στο προηγούμενο παράδειγμα, αυτό πραγματοποιείται, με τη συνάρτηση του χρήστη sanit_input. Περισσότερα για τη λειτουργία της θα αναφερθούν σε παρακάτω ενότητα.

### 5.2.15 Αποθήκευση δεδομένων

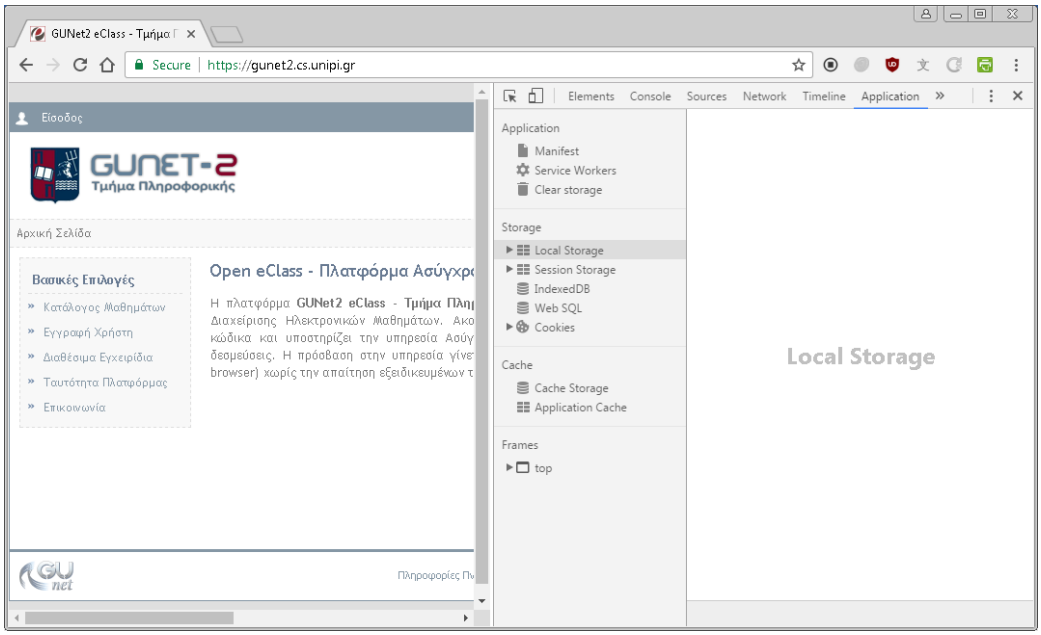
Όλο και περισσότερο γίνεται φανερή η ανάγκη για δυνατότητα αποθήκευσης πληροφοριών. Οι πληροφορίες αυτές είναι δυνατόν να χρησιμοποιούνται από την ίδια την εφαρμογή για την ορθότερη λειτουργία της, είτε να τις χρειάζεται ο εξυπηρετητής ιστού ώστε να ανταπεξέλθει καλύτερα στο έργο του. Το περιεχόμενο των πληροφοριών μπορεί να αφορά προσωπικά στοιχεία του χρήστη είτε ανώνυμα δεδομένα του περιβάλλοντος εργασίας, π.χ. ποιος φυλλομετρητή χρησιμοποιείται, ποιο λειτουργικό σύστημα κλπ.

Η διαδικασία της αποθήκευσης μπορεί να διαφοροποιηθεί με ποικίλους τρόπους. Μπορεί να διαφοροποιηθεί ανάλογα με το 'που' θα γίνει αποθήκευση. Η αποθήκευση μπορεί να πραγματοποιηθεί είτε τοπικά στον υπολογιστή του επισκέπτη είτε απομακρυσμένα σε κάποια βάση δεδομένων, κάποιο αρχείο στον υπολογιστή του εξυπηρετητή. Η επιλογή του 'τί' θα αποθηκευτεί επηρεάζει την υλοποίηση της διαδικασίας. Για παράδειγμα, ευαίσθητα δεδομένα, όπως συνθηματικά, αποθηκεύονται πάντα στην πλευρά του εξυπηρετητή και, κατά κύριο λόγο, σε κάποια βάση δεδομένων. Η επιλογή του 'πώς' θα πραγματοποιηθεί η αποθήκευση, επιβάλει, τους δικούς της περιορισμούς. Τα δεδομένα μπορούν να αποθηκευτούν συνολικά, ως αρχεία περιεχομένου (π.χ. ήχος, αρχεία μορφοποίησης .css), ή με τη μορφή συσχετιζόμενου πίνακα, δηλαδή με τη μορφή κλειδί=>τιμή.
Οπουδήποτε εμπλέκεται η έννοια της αποθήκευσης πληροφοριών που αφορούν τον χρήστη μιας ιστοσελίδας, έστω και αν αυτές είναι ανώνυμες, χρειάζεται ιδιαίτερη προσοχή. Για παράδειγμα, από το Μάη του 2012, οποιαδήποτε ιστοσελίδα, η οποία είναι διαθέσιμη στο ευρωπαϊκό κοινό, πρέπει να συμμορφώνεται με την οδηγία της Ευρωπαϊκής Ένωσης EU E-Privacy. Νέοι νόμοι μπήκαν σε ισχύς το 2011, οι οποίοι εμποδίζουν την αποθήκευση προσωπικών πληροφοριών στον υπολογιστή του χρήστη, χωρίς την έγκριση και άδεια του. Εφόσον μια εφαρμογή χρησιμοποιεί αποθήκευση στην πλευρά του χρήστη θα πρέπει:

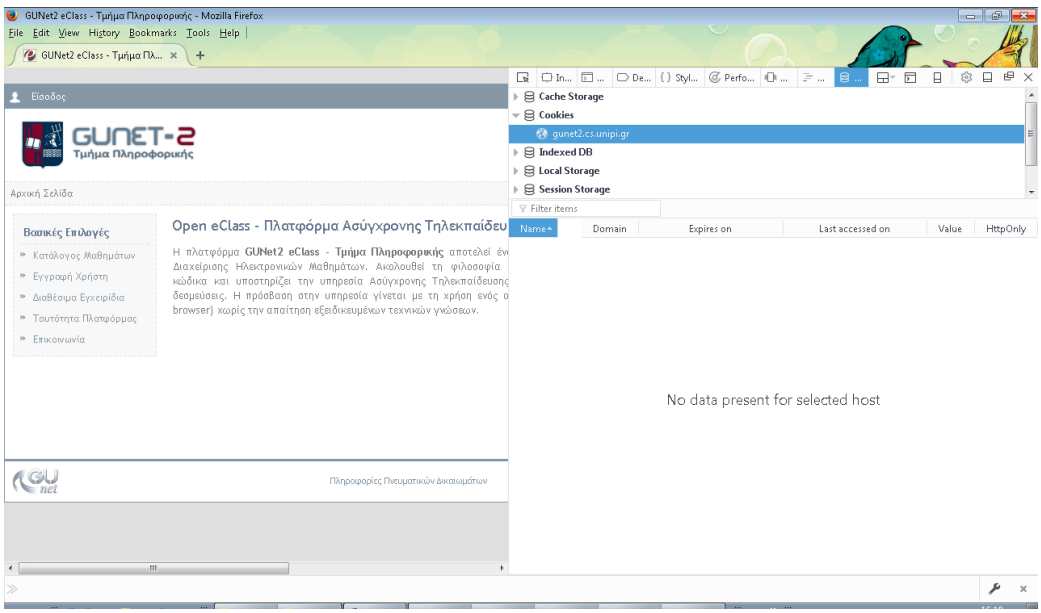
- ✓ Να ενημερωθεί ο χρήστης για τις χρησιμοποιούμενες τεχνικές αποθήκευσης.
- ✓ Να αιτιολογηθούν οι λόγοι που επιβάλουν τη χρήση αυτών των τεχνικών.
- ✓ Να εξασφαλιστεί η συγκατάθεση του χρήστη πριν τη χρήση αυτών των τεχνικών και είναι εφικτή και η υπαναχώρηση τους οποιαδήποτε στιγμή το επιθυμούν.

Η επισκόπηση των προσωρινά αποθηκευμένων πληροφοριών μπορεί να πραγματοποιηθεί μέσα από τον φυλλομετρητή του επισκέπτη. Στις εικόνες XXX

και XXX παρουσιάζονται τα προσωρινά αποθηκευμένα δεδομένα στους φυλλομετρητές chrome και firefox. Η προβολή πραγματοποιείται επιλέγοντας με το ποντίκι δεξί κλικ και inspect.




**Εικόνα 5.6 – Προσωρινά αποθηκευμένες πληροφορίες. Chrome.**



**Εικόνα 5.7 – Προσωρινά αποθηκευμένες πληροφορίες. Firefox.**

**<u>Στην πλευρά του πελάτη</u>**

Όταν η αποθήκευση λαμβάνει χώρα στον τοπικό υπολογιστή του επισκέπτη εφαρμόζεται χρησιμοποιώντας τεχνικές της javascript, όπως το API (application programming interface) Application Cache, Cookies, WebStorage το οποίο περιλαμβάνει το LocalStorage, SessionStorage WebDatabase.

***Προσωρινή αποθήκευση περιεχομένου (cashe)***

Ένας από τους τομείς, στον οποίο έχει εστιάσει η HTML5 είναι η διαθεσιμότητα των εφαρμογών εκτός σύνδεσης. Μια εφαρμογή Διαδικτύου χωρίς σύνδεση εξακολουθεί να λειτουργεί ακόμα και όταν δεν υπάρχει διαθέσιμη σύνδεση Internet. Η ανάγκη αυτής της λειτουργίας προέκυψε από την επιθυμία του συναγωνισμού με τις παραδοσιακές εφαρμογές πελάτη-εξυπηρετητή, οι οποίες μπορούν πάντα να χρησιμοποιηθούν, εφόσον η συσκευή είναι ενεργή. Αυτό σε συνδυασμό με το γεγονός ότι πολλές φορές η σύνδεση με το Internet μπορεί να είναι ασταθής οδήγησαν στον ορισμό του javascript API (application programming interface) Application Cache της HTML5.

Η ιδέα της αποθήκευσης περιεχομένου ιστοσελίδων τοπικά από το φυλλομετρητή (Browser Cache) δεν είναι καινούρια. Όλοι οι φυλλομετρητές μπορούν να αποθηκεύσουν ιστοσελίδες, το περιεχόμενο τους σε συνδυασμό με τα αρχεία βίντεο, ήχου, εικόνων, ή/και άλλων συνοδευτικών αρχείων (π.χ. .css, .js,) που χρησιμοποιούνται. Στην περίπτωση αυτή ο φυλλομετρητής αποφασίζει το τι και το που θα αποθηκεύσει τις απαιτούμενες πληροφορίες.

Η διαφορά της προσέγγισης, η οποία μπορεί να πραγματοποιηθεί μέσα από την HTML5 είναι η 'εξυπνότερη' διαχείριση. Μέχρι την HTML 4 ο χρήστης θα έπρεπε να αποθηκεύει προσωρινά κάθε ιστοσελίδα μεμονωμένα. Στην HTML5 μπορεί ο προγραμματιστής, κατά τη διάρκεια της ανάπτυξης της εφαρμογής, να ορίσει ποια αρχεία πρέπει να μπορούν να αποθηκεύονται προσωρινά. Αυτό μπορεί να γίνει είτε σε επίπεδο σελίδων εφαρμογής είτε σε επίπεδο σελίδας, δηλαδή μέσα στην ίδια σελίδα κάποια αρχεία να αποθηκεύονται προσωρινά και κάποια άλλα όχι. Κατά συνέπεια ο ιστοτόπος συνεχίζει να λειτουργεί κανονικά και όταν γίνεται ανανέωση και δεν υπάρχει σύνδεση Διαδικτύου.

Αυτού του είδους η προσωρινή αποθήκευση έχει διάφορα πλεονεκτήματα, όπως:

- ✓ Διαθεσιμότητα χωρίς σύνδεση: Ο χρήστης είναι σε θέση να περιηγηθεί στον ιστοτόπο, ακόμη και όταν δεν είναι συνδεδεμένος στο Διαδίκτυο.
- ✓ Ταχύτητα: Τα αρχεία, τα οποία είναι αποθηκευμένα προσωρινά τοπικά χρησιμοποιούνται πολύ πιο γρήγορα. Για παράδειγμα τα αρχεία κανόνων μορφοποίησης (.css) χρησιμοποιούνται από πολλαπλές σελίδες σε έναν ιστοτόπο. Έτσι ενώ κατά την πρώτη επίσκεψη στην ιστοσελίδα, θα

χρειαστεί κάποιος χρόνος ώστε να κατέβει τοπικά το αρχείο μορφοποίησης, εντούτοις αυτός ο χρόνος ελαχιστοποιείται για κάθε μελλοντική επίσκεψη στην ίδια σελίδα, καθώς και για κάθε επίσκεψη ιστοσελίδας του ίδιου ιστοτόπου.

- ✓ Μείωση φόρτου εξυπηρετητή και δικτύου: Κάθε φορά που πραγματοποιείται επίσκεψη σε κάποια ιστοσελίδα, ο φυλλομετρητής 'ζητά' από τον εξυπηρετητή να ελέγξει εάν έχει διαφοροποιηθεί το περιεχόμενο των αρχείων τα οποία είναι αποθηκευμένα προσωρινά. Στην περίπτωση όπου δεν έχει διαπιστωθεί κάποια διαφοροποίηση δεν 'κατεβαίνουν' τοπικά τα αρχεία αυτά. Με αυτή τη διαδικασία μειώνεται σημαντικά ο φόρτος του εξυπηρετητή, καθώς η 'κίνηση' του δικτύου.
- ✓ Καλύτερη διαχείριση αποθηκευτικού χώρου: Οι φυλομετρητές κινητών συσκευών έχουν, τυπικά, μικρό προσωρινό αποθηκευτικό χώρο. Ένα μέσο μέγεθος σελίδας για φορητές συσκευές είναι 549KB, ως εκ τούτου, ο αποθηκευτικός χώρος του φυλλομετρητή θα πρέπει να ενημερωθεί πλήρως μετά από 8 επιτυχημένες επισκέψεις στη σελίδα. Η HTML5 μέσω του Application Cache παρέχει επιπλέον έλεγχο των πόρων που αποθηκεύονται προσωρινά. Επιπλέον χρησιμοποιεί διαφορετικό χώρο προσωρινής αποθήκευσης από τον φυλλομετρητή, το οποίο σημαίνει ότι δεν υπόκεινται στους ίδιους περιορισμούς μεγέθους.
- ✓ Ασφάλεια: Στο αρχείο δήλωσης (manifest file) περιέχονται οι πόροι της εφαρμογής που 'κατεβαίνουν' σε κάθε επίσκεψή της. Εάν το αρχείο αυτό είναι 'κλειστό', τότε μόνο οι πόροι, οι οποίοι περιγράφονται στο αρχείο μπορούν να 'κατεβούν' τοπικά. Εμποδίζοντας, με αυτόν τον τρόπο, οποιαδήποτε άλλη πηγή προσπαθήσει να αποθηκεύσει οτιδήποτε τοπικά.

Προηγούμενα αναφέρθηκαν τα πλεονεκτήματα χρήσης της προσωρινής τοπικής αποθήκευσης, μέσω του Application Cache της HTML5, υπάρχουν, ωστόσο, και ορισμένα μειονεκτήματα, όπως:

- ✓ Διπλός χρόνος ανανέωσης: Αυτό αποτελεί και το μεγαλύτερο μειονέκτημα στη χρήση του αρχείου δήλωσης. Εάν το περιεχόμενο της ιστοσελίδας έχει διαφοροποιηθεί, ο χρήστης θα δει το προσωρινά αποθηκευμένο περιεχόμενο, έως ότου αυτό ενημερωθεί. Εντούτοις τα προσωρινά αποθηκευμένα αρχεία ενημερώνονται κατά την επίσκεψη στην ιστοσελίδα. Αυτό τελικά σημαίνει ότι ο χρήστης θα δει το ενημερωμένο περιεχόμενο κατά τη δεύτερη φορά που θα επισκεφτεί την ιστοσελίδα. Δηλαδή κατά την πρώτη επίσκεψη ο χρήστης βλέπει την ιστοσελίδα με το παλιό περιεχόμενο, ενώ ταυτόχρονα και με τρόπο αδιαφανή, επικαιροποιείται το περιεχόμενο. Το νέο επικαιροποιημένο περιεχόμενο θα είναι διαθέσιμο στο χρήστη, κατά τη δεύτερη επίσκεψη.

- ✓ Εξατομικευμένο: Ο φυλλομετρητής συνεχίζει να εμφανίζει το αρχείο, το οποίο είναι αποθηκευμένο τοπικά, ακόμα και όταν αυτό έχει διαφοροποιηθεί στον εξυπηρετητή. Κατά την περίπτωση όπου χρειάζεται να επικαιροποιηθεί κάποιο μεμονωμένο αρχείο, αυτό πραγματοποιείται με την ανανέωση όλων των αρχείων. Οποιαδήποτε αλλαγή πραγματοποιηθεί στο αρχείο δήλωσης προκαλεί την ανανέωση όλων των περιγραφόμενων αρχείων.
- ✓ Ασύγχρονη αποθήκευση: Οι πόροι της εφαρμογής που περιγράφονται στο αρχείο δήλωσης αποθηκεύονται τοπικά, κατά την αποθήκευση του αρχείου δήλωσης. Ως εκ τούτου, θα γίνει λήψη πόρων που μπορεί να μην απαιτούνται, όπως JavaScript ή μια εικόνα. Αυτό σημαίνει ότι είναι να ελέγχετε δυσκολότερα η σειρά που έχουν αποθηκευτεί τοπικά οι πόροι.

Ο μηχανισμός για να διασφαλιστεί ότι η ιστοσελίδα είναι διαθέσιμη για τον χρήστη, ακόμη και όταν δεν υφίσταται σύνδεση με το Διαδίκτυο, είναι απλός.
Χρειάζεται να συμπληρωθεί το χαρακτηριστικό `manifest` στο στοιχείο `html`, όπως παρουσιάζεται στο παρακάτω παράδειγμα κώδικα.
Το χαρακτηριστικό περιλαμβάνει ένα URI για το αρχείο δηλώσεων, το οποίο περιέχει τους κανόνες για την προσωρινή αποθήκευση.

```
<!DOCTYPE HTML>
      <html manifest="manifest.appcache">
      ...
```

***Σημείωση****: Η προτεινόμενη επέκταση των αρχείων δήλωσης (manifest files) είναι η ".appcache".*

Παρακάτω παρουσιάζεται το περιεχόμενο ενός τυπικού αρχείου δήλωσης.

```
# εισαγωγή σχολίων στο αρχείο manifest.
CACHE MANIFEST
index.html
stylesheet.css
images/masthead.png
scripts/misc.js
NETWORK:
search.php
login.php
/api
```

```
FALLBACK:
images/dynamic.php static_image.png
```

Όπως φαίνεται τα αρχεία αυτού του τύπου περιέχουν τρείς ενότητες: CACHE MANIFEST, NETWORK, FALLBACK.
Τα αρχεία που αναφέρονται στην ενότητα CACHE MANIFEST, θα αποθηκεύονται προσωρινά κατά την πρώτη επίσκεψη στην ιστοσελίδα, ενώ αυτά που αναφέρονται στην ενότητα NETWORK, δεν θα αποθηκεύονται ποτέ και, επιπλέον, χρειάζονται την παρουσία ενεργής σύνδεσης Διαδικτύου.
Στην ενότητα FALLBACK, αναφέρονται οι εναλλακτικοί πόροι της εφαρμογής. Εάν ο φυλλομετρητής δεν είναι δυνατό να ανακτήσει το αρχικό περιεχόμενο, θα χρησιμοποιήσει την εναλλακτική πηγή. Στο παραπάνω παράδειγμα, εμφανίζεται μια στατική εικόνα, στην περίπτωση που η δυναμική δεν είναι διαθέσιμη.
Η τελευταία γραμμή στην ενότητα NETWORK περιέχει τη διαδρομή προς ένα φάκελο, ώστε να εξασφαλίζεται ότι οι αιτήσεις για τη φόρτωση πόρων που περιλαμβάνονται μέσα στο φάκελο /api, θα παρακάμψουν τη προσωρινή αποθήκευση και θα παρουσιάσουν τον πόρο, ο οποίος βρίσκεται στον εξυπηρετητή.
Στο αρχείο δήλωσης οι γραμμές σχολίων ορίζονται να ξεκινάν με το σύμβολο #. Τα σχόλια εκτός από την προφανή χρήση της αναγνωσιμότητάς του κώδικα, έχουν και άλλη χρήση. Επειδή τα αρχεία θα επικαιροποιηθούν μόνο έναν το αρχείο δήλωσης έχει διαφοροποιηθεί, ο χρήστης θα συνεχίζει να βλέπει το παρωχημένο περιεχόμενο. Εάν χρειάζεται να επιβληθεί επικαιροποίηση, αυτή μπορεί να πραγματοποιηθεί αλλάζοντας μέρος του αρχείου δήλωσης, π.χ μεταβάλλοντας τον αριθμό έκδοσης του αρχείου η οποία περιέχεται σε γραμμή σχολίων.
Μόλις το περιεχόμενο αποθηκευτεί τοπικά προσωρινά, επικαιροποιείται μόνο εφόσον πραγματοποιηθεί κάποιο από τα παρακάτω:

- ✓ Όταν ο χρήστης διαγράψει τα προσωρινά αποθηκευμένα δεδομένα του φυλλομετρητή
- ✓ Το αρχείο δήλωσης έχει διαφοροποιηθεί.
- ✓ Τα προσωρινά αποθηκευμένα δεδομένα έχουν επικαιροποιηθεί μέσα από την εφαρμογή προγραμματιστικά.

Παρακάτω εξετάζονται τεχνικές, οι οποίες αποθηκεύουν τις πληροφορίες με τη μορφή συσχετιζόμενου πίνακα.

### *Cookies*

Ένας τρόπος που έχει χρησιμοποιηθεί κατά κόρον στο παρελθόν για την αποθήκευση πληροφοριών στον τοπικό υπολογιστή του χρήστη, αποτελεί και η

χρήση των cookies. Τα cookies αποτελούν μικρές ‘ποσότητες’ δεδομένων τα οποία δημιουργούνται από την εφαρμογή και αποθηκεύονται στο φυλλομετρητή του χρήστη. Χρησιμοποιήθηκαν, αρχικά, για την επικοινωνία του εξυπηρετητή ιστού και του φυλλομετρητή του χρήστη. Η δημιουργία τους προέκυψε από την ανάγκη να αποθηκεύονται τοπικά πληροφορίες, για τον χρήστη και την επίσκεψή του. Με τον τρόπο αυτό, ο ιστότοπος θυμάται τις ενέργειές και τις προτιμήσεις (όπως κωδικός σύνδεσης, γλώσσα, μέγεθος γραμματοσειράς και άλλες προτιμήσεις απεικόνισης) για ένα χρονικό διάστημα, κι έτσι δεν χρειάζεται να εισάγονται οι προτιμήσεις αυτές κάθε φορά που επισκέπτεται ο χρήστης τον ιστότοπο.

Τα cookies μπορεί να περιέχουν σχεδόν οποιαδήποτε αλφαριθμητικές πληροφορίες (εφ 'όσον αυτό είναι κάτω των 4 KB) και μπορούν να ανακτηθούν από τον υπολογιστή και να επιστραφούν στον εξυπηρετητή. Αποτελούν μια τεχνική, η οποία υποστηρίζεται από όλους σχεδόν τους φυλλομετρητές.

Εξαιτίας της φύσης των πληροφοριών που περιέχουν και τις συνέπειές τους στην ιδιωτική ζωή, τα cookies μπορούν να διαβαστούν μόνο από τον τομέα που τα εξέδωσε. Με άλλα λόγια, εάν ένα cookie εκδίδεται από, για παράδειγμα, unipi.gr, μπορεί να ανακτάται μόνο από τον εξυπηρετητή ιστού χρησιμοποιώντας αυτόν τον τομέα. Αυτό αποτρέπει άλλους δικτυακούς τόπους από να αποκτήσουν πρόσβαση σε πληροφορίες για τις οποίες δεν έχουν τα κατάλληλα δικαιώματα.

Συνήθως ο τομέας ενός cookie ταυτίζεται με την περιγραφή που εμφανίζεται στη γραμμή διεύθυνσης του φυλλομετρητή. Αυτό ονομάζεται πρώτου μέρους (first-party) cookie. Ένα cookie τρίτου μέρους (third-party), όμως, ανήκει σε διαφορετικό τομέα από εκείνον που εμφανίζεται στη γραμμή διευθύνσεων. Αυτό το είδος cookie εμφανίζεται, συνήθως, σε ιστοσελίδες οι οποίες διαθέτουν περιεχόμενο από εξωτερικές ιστοσελίδες, όπως διαφημίσεις banner. Αυτός ο τρόπος παρέχει τη δυνατότητα παρακολούθησης του ιστορικού περιήγησης του χρήστη. Χρησιμοποιείται, δε, συχνά από τις διαφημιστικές εταιρείες ώστε να παρουσιάσουν σε κάθε χρήστη σχετικές/προσαρμοσμένες διαφημίσεις.Εξαιτίας αυτού, οι περισσότεροι φυλλομετρητες επιτρέπουν στους χρήστες να απενεργοποιήσουν τα cookies από τον τρέχοντα τομέα του εξυπηρετητή, τομέα τρίτου μέρους, ή και τα δύο.

Τα cookies ανταλλάσσονται κατά τη διάρκεια της μεταφοράς των επικεφαλίδων του πρωτοκόλλου HTTP και πριν από την αποστολή οποιαδήποτε τμήματος HTML μιας ιστοσελίδας. Είναι αδύνατο να σταλεί ένα cookie από τη στιγμή που έχει μεταφερθεί τμήμα HTML της ιστοσελίδας. Ως εκ τούτου, είναι σημαντικός ο προσεκτικός σχεδιασμός της χρήσης των cookie.

Η διαχείριση των cookies είναι εφικτή μέσα από τεχνολογίες στην πλευρά του επισκέπτη, π.χ. με javascript, εντούτοις στη συνέχεια της ενότητας θα

εξετάσουμε τον ορισμό, την πρόσβαση και την καταστροφή τους με τη χρήση της γλώσσας PHP.

***Σημείωση**: Ο ορισμός ενός cookie με τη χρήση της γλώσσα javascript πραγματοποιείται ως εξής:*

```
document.cookie = "key1=value1;key2=value2;expires=date";
```

*Η παρακάτω συνάρτηση ορίζει την πρόσβαση τα cookies μέσω javascript*

```
function ReadCookie()
      {
        var allcookies = document.cookie;
        document.write ("All Cookies : " + allcookies );

        // Αποθήκευση όλων των cookies σε πίνακα
        cookiearray = allcookies.split(';');

        // Διαχωρισμός
        for(var i=0; i<cookiearray.length; i++){
          name = cookiearray[i].split('=')[0];
          value = cookiearray[i].split('=')[1];
          document.write ("Το cookie είναι : " + name + " και η τιμή του: " + value);
        }
      }
```

*Τέλος η διαγραφή/καταστροφή ενός cookie πραγματοποιείται με τον καθορισμό της ημερομηνίας λήξης σε κάποια στιγμή στο παρελθόν. Για παράδειγμα ο παρακάτω κώδικας:*

```
function WriteCookie()
      {
        var now = new Date();
        now.setMonth( now.getMonth() - 1 );
        cookievalue = escape(document.myform.customer.value) + ";"

        document.cookie="name=" + cookievalue;
        document.cookie = "expires=" + now.toUTCString() + ";"
        document.write("Setting Cookies : " + "name=" + cookievalue );
      }
```

Ο ορισμός ενός cookie μέσα από την PHP είναι μια σχετικά απλή διαδικασία και πραγματοποιείται με τη χρήση της έτοιμης συνάρτησης `setcookie`. Εφόσον δεν έχει πραγματοποιηθεί μεταφορά οποιουδήποτε τμήματος κώδικα HTML, μπορεί να δημιουργηθεί ένα cookie, όπως φαίνεται στον παρακάτω κώδικα:

```
setcookie(name, value, expire, path, domain, secure,
httponly);
```

Η συνάρτηση μπορεί να δεχθεί τις παραμέτρους, οι οποίες περιγράφονται στον πίνακα XXX.

**Πίνακας 5.8 Παράμετροι της συνάρτησης setcookie**

| **Παράμετρος** | **Περιγραφή** |
|---|---|
| name | Το όνομα του cookie. Κάθε μελλοντική αναφορά του εξυπηρετητή σε αυτό το cookie θα πραγματοποιείται με τη χρήση αυτού του αναγνωριστικού. |
| value | Η τιμή/περιεχόμενο του. Μπορεί να είναι έως 4 KB ανά cookie και οποιοδήποτε αλφαριθμητικό κείμενο. Συνηθέστερες τιμές είναι το username και η ημερομηνία τελευταία επίσκεψης. |
| **Προαιρετικές παράμετροι** | |
| expire | Μια αριθμητική τιμή σε δευτερόλεπτα, αρχίζοντας από 00:00:00 GMT την 1η Ιανουαρίου 1970. Μετά από αυτή τη χρονική στιγμή το cookie είναι απροσπέλαστο. Εάν δεν οριστεί, αυτή η παράμετρος, τότε ως λήξη ορίζεται το κλείσιμο του παραθύρου του φυλλομτρητή. |
| path | Καθορίζει τους καταλόγους, για τους οποίους είναι έγκυρο το cookie. Μια απλή / (κάθετος) επιτρέπει στο cookie να ισχύει για όλους τους καταλόγους. Η προκαθορισμένη τιμή είναι ο τρέχον κατάλογος, μέσα στον οποίο έχει δημιουργηθεί το cookie. |
| domain | Ο τομέας του Διαδικτύου του cookie. Για παράδειγμα εάν αυτό είναι .unipi.gr τότε το cookie είναι διαθέσιμο για: www.unipi.gr, images.unipi.gr. Εάν, για παράδειγμα έχει οριστεί για το images.unipi.gr τότε είναι διαθέσιμο για τους π.χ. sub.images.unipi.gr, αλλά όχι και για τον www.unipi.gr. |
| secure | Καθορίζει τον τρόπο μεταφοράς του cookie πάνω από το πρωτόκολλο επικοινωνίας http ή https. Εφόσον είναι ‘αληθής’ (1), η μεταφορά θα πρέπει να συμβεί μόνο με https. Η προεπιλεγμένη τιμή είναι ‘ψευδής’ (0) . |
| httponly | Ισχύει από την έκδοση PHP 5.2.0 και μετέπειτα. Καθορίζει εάν θα είναι προσβάσιμο με στις τεχνολογίες στην πλευρά του επισκέπτη, π.χ. javascript. Εάν είναι ‘αληθής’ τότε το cookie είναι απροσπέλαστο στην javascript. Η προεπιλεγμένη τιμή |

| | |
|---|---|
| | είναι ‘ψευδής’. Δεν υποστηρίζεται από όλους τους φυλλομετρητές. |

Για παράδειγμα ο παρακάτω κώδικας δημιουργεί ένα cookie, με τις τιμές: name =>username, value=>rosa, domain=>ολόκληρος ο τομέας, expire=> θα λήξει σε επτά ημέρες από τη στιγμή της δημιουργία του.

```
setcookie('username', 'rosa', time() + 60 * 60 * 24 * 7,
'/');
```

Εφόσον έχει δημιουργηθεί κάποιο cookie δεν μπορεί να διαβαστεί/προσπελαστεί η τιμή του, παρά μόνον στην περίπτωση όπου έχει πραγματοποιηθεί ανανέωση σελίδας, οπότε και ο φυλλοετρητής θα το αποστείλει πίσω στον εξυπηρετητή. Όπως έχει προαναφερθεί όλα τα cookie περιέχονται στην υπερκαθολική μεταβλητή πίνακα $_COOKIE. Οπότε όταν χρειαστεί να προσπελαστεί κάποιο cookie απλά ελέγχεται η αντίστοιχη θέση του πίνακα $_COOKIE. Για παράδειγμα ο παρακάτω κώδικας ελέγχει εάν έχει οριστεί (και έχει κάποια τιμή) το cookie με όνομα ‘username’ και στη συνέχεια αποθηκεύει την τιμή του στην μεταβλητή $username.

```
if (isset($_COOKIE['username'])) $username =
$_COOKIE['username'];
```

Η καταστροφή ενός cookie περιλαμβάνει τον ορισμό της παραμέτρου λήξης του σε κάποια παρελθοντική στιγμή. Αξίζει να σημειωθεί ότι αν υπάρχουν διαφοροποιήσεις στις τιμές των υπολοίπων παραμέτρων (εκτός της λήξης) σε σχέση με αυτές που χρησιμοποιήθηκαν κατά τη δημιουργία του, τότε η διαγραφή του θα αποτύχει. Κατά συνέπεια, για την καταστροφή του cookie που δημιουργήθηκε ανωτέρω, στην ενότητα αυτή, θα χρησιμοποιηθεί ο παρακάτω κώδικας, ο οποίος ορίζει ότι τη λήξη πριν από ένα μήνα.

```
setcookie('username', 'rosa', time() - 2592000, '/');
```

***Σημείωση****: Αποτελεί καλή πρακτική ο ορισμός της λήξης ενός cookie να είναι αρκετά ‘πίσω’, στο παρελθόν, ώστε να αποφεύγονται περαιτέρω προβλήματα στην περίπτωση όπου ο υπολογιστής του χρήστη δεν έχει σωστά ρυθμισμένη την ημερομηνία και ώρα.*

## 5.3 CGI Μπορεί και να φύγει

Τα σενάρια **CGI** (Common Gateway Interface) αποτελούν μία μέθοδο προγραμματισμού για το Διαδίκτυο. Με τη χρήση των σεναρίων CGI ένας υπολογιστής δικτύου μπορεί να λάβει τα στοιχεία από (ή να στείλει στοιχεία σε)

βάσεις δεδομένων, έγγραφα και άλλα προγράμματα και να παρουσιάσει τα στοιχεία αυτά σε ιστοσελίδες του Διαδικτύου.

Κανονικά όταν ένας φυλλομετρητής ανατρέχει σε ένα URL συμβαίνουν τα ακόλουθα. Πρώτα ο υπολογιστής έρχεται σε επαφή με τον εξυπηρετητή παγκόσμιου Ιστού, ο οποίος φιλοξενεί την ιστοσελίδα με το αντίστοιχο URL. Έπειτα ο εξυπηρετητής εξετάζει το όνομα του αρχείου που ζητείται από τον υπολογιστή και το στέλνει. Τέλος, ο φυλλομετρητής του τοπικού υπολογιστή παρουσιάζει το αρχείο στην κατάλληλη μορφή.

Είναι δυνατό όμως, να μην επιστραφεί το αρχείο στο οποίο «δείχνει» το URL, αλλά να εκτελεστεί σαν ένα οποιοδήποτε πρόγραμμα και να επιστραφούν (στον αρχικό υπολογιστή) τα αποτελέσματα της εκτέλεσης αυτής. Αυτή η λειτουργία καλείται Common Gateway Interface ή CGI. Τα προγράμματα αυτά καλούνται σενάρια CGI. Τα σενάρια **CGI** μπορούν να γραφτούν σε μία πληθώρα γλωσσών προγραμματισμού, με πιο δημοφιλή την **Perl.**

### 5.4.1 Βασικοί κανόνες κατασκευής σεναρίων CGI σε Perl

Τοποθετούμε υποχρεωτικά την παρακάτω γραμμή στην κορυφή του σεναρίου Perl που θα γράψουμε:

```
#!/usr/local/bin/perl
```

Με αυτή τη γραμμή λέμε στον εξυπηρετητή, που θα εκτελέσει το σενάριο, ότι ο κώδικας που ακολουθεί είναι σε γλώσσα Perl. Ο εξυπηρετητής πρέπει να γνωρίζει τι είναι αυτό που του ζητείται να εκτελέσει. Για το λόγο αυτό δεν πρέπει να ξεχνάμε να ξεκινάμε τον κώδικα μας με αυτή τη γραμμή.

Στο τέλος κάθε γραμμής πρέπει να βάζουμε ένα ερωτηματικό (;). Αν

δεν το βάλουμε, τότε το πρόγραμμα δεν θα εκτελεστεί ποτέ.

Βάζουμε σαφή σχόλια σε κάθε γραμμή κώδικα που γράφουμε, το οποίο θεωρείται γενικά απαραίτητο σε κάθε πρόγραμμα που δημιουργούμε. Κάθε γραμμή σχολίου ξεκινάει με το χαρακτήρα "#", που δηλώνει ότι πρόκειται για σχόλιο. Εξαίρεση αποτελεί η γραμμή που δηλώνει την γλώσσα συγγραφής του σεναρίου, δηλαδή η `#!/usr/local/bin/perl`. Τα παρακάτω αποτελούν μερικά παραδείγματα σχολίων:

```
# Με τον παρακάτω κώδικα θα κατασκευάσουμε μία μεταβλητή,
η τιμή # της οποίας θα αποθηκεύεται σε κάποιο αρχείο log.
```

```
my $in{'logdata'} = <<END;
$in{'name'}
$in{'formelement1'}    |    $in{'radiobutton2'}
$in{'checkbox1'}       |    $in{'checkbox2'}


END
```

Με τα σχόλια ο κώδικας γίνεται καλογραμμένος και μπορεί να επεξεργαστεί πιο εύκολα από κάποιον που δε τον έχει γράψει και προσπαθεί να τον καταλάβει και, ίσως, να τον τροποποιήσει.

Ο κώδικας πρέπει να είναι καθαρός, για να είναι συντηρήσιμος. Ο παρακάτω κώδικας δεν είναι λάθος, αλλά είναι όμως κακογραμμένος και δύσκολος στην ανάγνωση και κατανόηση.

```
for ($i=0;$i<=$#max;$i++){if ($max[$i]~=/hello there/g;)
{print  "bad code";}else { print "bye";}}
```

Ακόμα και με σχόλια θα έπαιρνε μέρες να καταλάβει ένα άτομο πως πρόκειται απλά για έναν βρόχο `for`, που τρέχει διαμέσου ενός πίνακα "max", ψάχνοντας για την φράση "hello there". Αν η φράση υπάρχει, τότε εκτυπώνεται η φράση "bad code" και αν δεν υπάρχει εκτυπώνεται η "bye". Ο παραπάνω κώδικας καλογραμμένος, φαίνεται ως εξής:

```
for ($i = 0; $i <= $#max; $i++) {
       if ($max[$i] ~= /hello there/g;) {
              print "bad code";
       }else {
              print "bye";
       }
}
```

Είναι χρήσιμο να δίνουμε στις μεταβλητές ονόματα που περιγράφουν αυτό που δηλώνουν. Στα ονόματα μεταβλητών απαγορεύονται τα κενά αλλά επιτρέπεται ο χαρακτήρας υπογράμμισης (_), για να ξεχωρίσουμε τις λέξεις. Π.χ. το $book_page_i_am_on είναι πιο κατανοητό από το $bookpageiamon.

Στα ονόματα των μεταβλητών μπορούμε να χρησιμοποιήσουμε όλα τα γράμματα (κεφαλαία και πεζά) της αλφαβήτου, αλλά όχι άλλους ειδικούς χαρακτήρες, όπως π.χ. !, @, #, $, %, ^, &, *, (, ), ~ κτλ.

Έχοντας δει τους βασικούς κανόνες, μπορούμε να προχωρήσουμε στην εκμάθηση των βασικών εντολών της **Perl.**

### 5.4.2 Οι μεταβλητές στην Perl

Οι μεταβλητές στην Perl ανήκουν σε έναν από τους παρακάτω τρεις τύπους:

- βαθμωτή (scalar),
- πίνακας (array), ή
- συσχετιζόμενος πίνακας (associative array).

**Βαθμωτές μεταβλητές**

Μία βαθμωτή μεταβλητή μπορεί να περιέχει αριθμούς, γράμματα, φράσεις, κτλ. Το σύμβολο του δολαρίου ($) προηγείται πάντα αυτής της μεταβλητής. Έτσι αν θελήσουμε να δώσουμε μία τιμή στην μεταβλητή $comp, γράφουμε:

Μία λέξη: `$comp = "word";`
Μία φράση: `$comp = "This is a phrase!";`
Έναν αριθμό: `$comp = 4;`

Ως τιμή μεταβλητής μπορεί επίσης να δοθεί ένα άθροισμα ή μία διαφορά ως εξής:

```
$comp= 4 + 2;
$comp= 4 - 2;
```

Η `$comp` θα ήταν ίση με 4 + 2 ή 4-2, αντίστοιχα.

Οι εκχωρήσεις σε μεταβλητές γίνονται πάντα από αριστερά στα δεξιά. Π.χ. η εντολή `4 + 2 = $comp;` είναι λάθος.

**Μεταβλητές τύπου πίνακα**

Μία μεταβλητή τύπου πίνακα κρατάει πολλές βαθμωτές μεταβλητές σε αριθμημένες θέσεις. Η πρόσθεση επιπλέον θέσεων μπορεί να γίνεται δυναμικά με αποτέλεσμα να αυξάνεται η μεταβλητή. Υπάρχει και η δυνατότητα μείωσης, αλλά αυτό απλά είναι χάσιμο χρόνου. Στις μεταβλητές τύπου πίνακα προηγείται το σύμβολο @. Όταν αναφερόμαστε σε μία μόνο θέση, μπορούμε να χρησιμοποιήσουμε το $. Μπορούμε να δηλώσουμε όσες θέσεις θέλουμε με τη μία εντολή γράφοντας:

```
@comp = ("1","2","one","hello world","learning Perl");
```

Για να αναφερθούμε σε κάθε θέση ξεχωριστά, χρησιμοποιούμε την κλήση με αριθμό (call by number): π.χ. στο παραπάνω παράδειγμα η `$comp[3]` είναι ίση με `"hello world"`, και η `$comp[1]` είναι ίση με `"2"`. Όπως παρατηρούμε

η Perl, όπως και η JavaScript ξεκινάει την αρίθμηση από το μηδέν. Γι' αυτό `$comp[0]` ισούται με `"1"`, `$comp[2]` με `"one"`, κτλ.

Μπορούμε να βρούμε πόσες θέσεις έχει ένας πίνακας ελέγχοντας την ενσωματωμένη μεταβλητή `$#comp`. Για παράδειγμα, ο αριθμός των θέσεων του πίνακα `@comp`, που ορίσαμε παραπάνω θα είναι 4. (Ξεκινάμε τη μέτρηση από 0, επομένως οι θέσεις είναι 5).

Για να δηλώσουμε μία θέση τη φορά γράφουμε, π.χ.:

```
$comp[0] = "2000000";
```

**Σημείωση:** Τα `$comp[0], $comp[30], ή $comp[οτιδήποτε]` δεν έχουν καμία σχέση με το $comp από το προηγούμενο παράδειγμα. Το μόνο που μοιράζονται είναι το ίδιο όνομα, που όμως δεν δημιουργεί κάποιο πρόβλημα εφόσον οι δύο μεταβλητές είναι διαφορετικού τύπου. Καλό είναι να μη δίνουμε σε διαφορετικές μεταβλητές ίδια ονόματα..

**Μεταβλητές τύπου σχετιζόμενου πίνακα**

Ο τελευταίος τύπος μεταβλητής είναι ο συσχετιζόμενος πίνακας. Τέτοιοι πίνακες χρησιμοποιούνται για την αποθήκευση συσχετιζόμενης πληροφορίας. Είναι πιο εύκολοι από τους απλούς πίνακες που είδαμε προηγουμένως, γιατί δεν χρειάζεται να θυμόμαστε τη θέση στην οποία έχει αποθηκευτεί κάποια πληροφορία. Το σύμβολο % προηγείται του ονόματος ενός συσχετιζόμενου πίνακα, αλλά όπως και στους άλλους πίνακες μπορούμε και εδώ να χρησιμοποιήσουμε το σύμβολο $ για να αναφερθούμε σε μία θέση ξεχωριστά. Για τον ορισμό ενός συσχετιζόμενου πίνακα γράφουμε:

```
%comp
=("song","track","lyrics","music","artist","band","year"
,"drink");
```

Κάθε τιμή σε αυτό το παράδειγμα είναι το κλειδί για την επόμενη τιμή. Έτσι, `$comp{'song'}` είναι `"track"` και `$comp{'music'}` είναι `"artist"`, κοκ. Για να ορίσουμε μία θέση τη φορά, γράφουμε:

```
$comp{'song'} = "Tuesday is gonne";
```

Για την επεξεργασία των μεταβλητών χρησιμοποιούμε τους τελεστές της Perl.

### 5.4.3 Τελεστές

Οι τελεστές χρησιμοποιούνται για πράξεις ανάμεσα στις μεταβλητές. Υπάρχουν τρεις διαφορετικοί τύποι τελεστών: εκχώρησης, σύγκρισης, και μαθηματικοί.

**Τελεστές εκχώρησης**

Οι τελεστές εκχώρησης (Assignment operators) δίνουν μία τιμή σε μία μεταβλητή. Οι τελεστές εκχώρησης της PERL παρουσιάζονται στον Πίνακα 5.2.1

**Πίνακας 5.9 Τελεστές εκχώρησης στην Perl**

| Τελεστής | Περιγραφή |
|---|---|
| = | Κάνει την τιμή της μεταβλητής στα αριστερά ίση με οτιδήποτε βρίσκεται στα δεξιά. |
| += | Προσθέτει την τιμή που είναι δεξιά στην τιμή αριστερά και κάνει την μεταβλητή στα αριστερά ίση με το αποτέλεσμα. |
| -= | Ίδιο με το παραπάνω, μόνο που γίνεται αφαίρεση. |

**Τελεστές σύγκρισης**

Οι τελεστές σύγκρισης (comparison operators) συγκρίνουν δύο τιμές και δίνουν μία τρίτη τιμή που εξαρτάται από το αποτέλεσμα της σύγκρισης. Οι τελεστές σύγκρισης παρουσιάζονται στον Πίνακα 5.2.2.

**Πίνακας 5.10 Τελεστές σύγκρισης στην Perl**

| Τελεστής | Περιγραφή |
|---|---|
| < | Επιστρέφει την τιμή ‘αληθής’ (true) αν η τιμή της μεταβλητής στα αριστερά είναι μικρότερη από αυτή στα δεξιά. Αλλιώς επιστρέφει την τιμή ‘ψευδής’ (false). |
| > | Επιστρέφει την τιμή ‘αληθής’ (true) αν η τιμή της μεταβλητής στα αριστερά είναι μεγαλύτερη από αυτή στα δεξιά. Αλλιώς επιστρέφει την τιμή ‘ψευδής’ (false). |
| >= | Επιστρέφει την τιμή ‘αληθής’ (true) αν η τιμή της μεταβλητής στα αριστερά είναι μεγαλύτερη ή ίση από αυτή στα δεξιά. Αλλιώς επιστρέφει την τιμή ‘ψευδής’ (false). |

| | |
|---|---|
| <= | Επιστρέφει την τιμή ‘αληθής’ (true) αν η τιμή της μεταβλητής στα αριστερά είναι μικρότερη ή ίση από αυτή στα δεξιά. Αλλιώς επιστρέφει την τιμή ‘ψευδής’ (false). |
| == | Επιστρέφει την τιμή ‘αληθής’ (true) αν οι τιμές και στις δύο πλευρές είναι ίσες; Αλλιώς επιστρέφει την τιμή ‘ψευδής’ (false). |
| eq | Το ίδιο με το παραπάνω, μόνο που συγκρίνει αλφαριθμητικά/κείμενο. |
| != | Επιστρέφει την τιμή ‘αληθής’ (true) αν η τιμή της μεταβλητής στα δεξιά δεν είναι ίση με την τιμή στα αριστερά. Αλλιώς επιστρέφει την τιμή ‘ψευδής’ (false) αν είναι ίσες. |
| ne | Το ίδιο με την παραπάνω. |

Οι τελεστές σύγκρισης είναι ιδανικοί για τις δηλώσεις `if…else` που είδαμε πιο πριν. Παρακάτω παρουσιάζεται ένα απλό παράδειγμα που συνδυάζει την χρήση των δηλώσεων `if…else` και των τελεστών.

```
#Τοποθέτηση της τιμής 5 στην μεταβλητή $comp.
$comp = 5;
# Σύγκριση της μεταβλητής $comp και αντίδραση σύμφωνα με το        #  αποτέλεσμα.
if ($comp < 5) {
       print "Good!";
} elsif ($comp >= 6) {
       print "Just ok";
} else {
       print "perfect!";
}
```

## Μαθηματικοί τελεστές

Οι μαθηματικοί τελεστές (mathematical operators) εκτελούν μαθηματικές πράξεις, όπως φαίνεται στον Πίνακα 5.2.3:

**Πίνακας 5.11 Μαθηματικοί τελεστές στην Perl**

| Τελεστής | Περιγραφή |
|---|---|
| * | Πολλαπλασιάζει δύο τιμές. |
| / | Διαιρεί δύο τιμές. |
| + | Προσθέτει δύο τιμές. |

| | |
|---|---|
| - | Αφαιρεί δύο τιμές. |
| ++ | Προσθέτει 1 στην τιμή στα αριστερά (έτσι αν $i είναι ίσο με 1, $i++ θα γίνει ίσο με 2). |
| -- | Αφαιρεί 1 από την τιμή στα αριστερά. |

Με εξαίρεση τους δύο τελευταίους μαθηματικούς τελεστές, οι υπόλοιποι δεν μπορεί να χρησιμοποιηθούν αν δεν χρησιμοποιήσουμε έναν τελεστή εκχώρησης για την αποθήκευση της τιμής. Έτσι θα πρέπει να γράψουμε για παράδειγμα:

```
$comp = 5 * 3;
```

### 5.4.4 Υπό συνθήκη εντολές στην Perl

Όταν θέλουμε ο κώδικας που γράφουμε να κάνει διάφορες ενέργειες, οι οποίες εξαρτώνται από συγκεκριμένες συνθήκες, χρησιμοποιούμε τις υπό συνθήκη εντολές ή εντολές διακλάδωσης. Στην Perl μια απλή εντολή διακλάδωσης είναι η `if…elseif…else`. Παρακάτω παρουσιάζεται η χρήση της εντολής αυτής.

```
if (statement) {
        task
} elsif (statement) {
        a different task
} else {
        a task if all else fails
}
```

### 5.4.5 Βρόχοι επανάληψης

Με την χρήση των βρόχων επανάληψης μπορούμε να επαναλάβουμε κάποιες γραμμές κώδικα χωρίς να χρειάζεται να τις ξαναγράφουμε. Οι βρόχοι επαναλαμβάνονται και σταματούν μόνο όταν ικανοποιούνται κάποιες παράμετροι. Όλοι οι βρόχοι πρέπει να περιέχουν κάποια εντολή τερματισμού. Για σωστό και εύκολα συντηρήσιμο πρόγραμμα πρέπει να μπορούμε να παρακολουθούμε τις μεταβλητές μας σε κάθε βήμα του βρόχου.

Η γενική μορφή ενός βρόχου είναι η παρακάτω:

```
εντολή (ελέγχου) {

        δηλώσεις;
```

```
}
```

Οι βασικοί τύποι βρόχων στην Perl είναι ο βρόχος `for`, ο βρόχος `while`, και ο βρόχος `foreach`.

**Ο βρόχος for**

Ο βρόχος `for` είναι στη μορφή του ο πιο πολύπλοκος όλων των τύπων των βρόχων. Η πολυπλοκότητα στον προγραμματισμό σημαίνει και περισσότερες δυνατότητες. Η μορφή ενός βρόχου `for` είναι:

```
for ($i = 0; $i <= $#comp; $i++) {
       print "$comp[$i]";
}
```

Στο βρόχο `for` ορίζουμε μία μεταβλητή που υπάρχει μόνο εντός του βρόχου και ορίζει τον αριθμό των επαναλήψεων. Στο παραπάνω παράδειγμα ορίζεται η μεταβλητή `$i` με αρχική τιμή ίση με `0`. Κάθε φορά που ο βρόχος επαναλαμβάνεται ελέγχει εάν το `$i` είναι μικρότερο ή ίσο από τον αριθμό των θέσεων του πίνακα `@comp` και αυξάνει την τιμή του `$i` κατά 1 (`$i +1`). Ο βρόχος σταματάει όταν το `$i` γίνει μεγαλύτερο από τον αριθμό των θέσεων του πίνακα `@comp`. Το πρόγραμμα εκτυπώνει τη θέση του πίνακα `@comp` που αντιπροσωπεύει το `$i`. Έτσι, στην πρώτη επανάληψη του βρόχου θα εκτυπωθεί `$comp[0]`. Στη δεύτερη, θα εκτυπωθεί `$comp[1]` και ο βρόχος θα σταματήσει όταν το `$i` ξεπεράσει τον αριθμό των στοιχείων του πίνακα.

**Ο βρόχος while**

Ο βρόχος `while` είναι παρόμοιος με τον `for`, μόνο που δεν είναι τόσο ανεξάρτητος. Ένα παράδειγμα ενός τέτοιου βρόχου είναι:

```
While ($comp ne "intel") {
       print "$comp";
       $comp = <NAMES>;
}
```

Αυτός ο κώδικας εκτελεί τις εντολές εντός του βρόχου όσο η μεταβλητή `$comp` δεν ισούται με "intel". Ο συγκεκριμένος αυτός τύπος βρόχου είναι λίγο επικίνδυνος από την άποψη ότι μπορεί να «πέσει» σε ατέρμονο loop, αν η μεταβλητή `$comp` δεν αλλάζει την τιμή της κάπου αλλού μέσα στο πρόγραμμα.

**Ο βρόχος foreach**

Οι βρόχοι `Foreach` είναι μία πιο χαλαρή έκδοση των βρόχων `for`.

```
foreach $slotnum (@comp) {
       print "$slotnum";
}
```

Στο παραπάνω παράδειγμα ο βρόχος θα εκτελεστεί τόσες φορές όσες είναι οι θέσεις του πίνακα `comp`. Σε κάθε επανάληψη ο βρόχος θα πάρει την τιμή και θα την εκχωρήσει στο `$slotnum` για να χρησιμοποιηθεί αργότερα. Έτσι ο πίνακας `@comp` θα ξεκινήσει με τη θέση 0 και θα φτάσει στην τελευταία θέση που υπάρχει. Ο βρόχος `foreach` είναι χρήσιμος στην προσπέλαση συσχετιζόμενων πινάκων μιας και οι θέσεις σε τέτοιους πίνακες δεν είναι αριθμημένες.

```
foreach $slotname (keys (%comp)) {
       print "$comp{$slotname}";
}
```

Ο κώδικας παίρνει κάθε τιμή από τον πίνακα `%comp`. Αυτό γίνεται με τη χρήση της ενσωματωμένης συνάρτησης `keys` της Perl, και στη συνέχεια εκτυπώνει την τιμή αυτή.

### 5.4.6 Εισαγωγή δεδομένων

Τα προγράμματα λαμβάνουν κάποια δεδομένα ως είσοδο, τα επεξεργάζονται και επιστρέφουν ως έξοδο τα αποτελέσματα της επεξεργασίας αυτής. Παρακάτω θα δούμε τον τρόπο με τον οποίο ένα πρόγραμμα αποκτά πρόσβαση στα δεδομένα εισόδου του και πώς μπορεί να εμφανίσει αποτελέσματα. Υπάρχουν δύο μεγάλες κατηγορίες εισόδου και εξόδου (input-output) δεδομένων: είσοδος-έξοδος από τον χρήστη και είσοδος-έξοδος από κάποιο αρχείο.

Στο Unix τα πάντα, ακόμα και οι συσκευές εισόδου / εξόδου (π.χ. οθόνη, εκτυπωτής κτλ.) θεωρούνται αρχεία και η Perl είναι μία γλώσσα προγραμματισμού προσαρμοσμένη σε αυτό το λειτουργικό σύστημα. Για να διαβάσει ένα πρόγραμμα κάποια δεδομένα που βρίσκονται σε ένα αρχείο θα πρέπει πρώτα να το ανοίξει, να διαβάσει τα δεδομένα και μετά να ξανακλείσει το αρχείο. Το αρχείο αναγνωρίζεται εσωτερικά από το πρόγραμμα ως κάποιο αντικείμενο που ονομάζεται «file handler». Κάθε αρχείο έχει το δικό του file handler. Έτσι για να εκτυπωθεί κάτι στην οθόνη, αφού αποτελεί αρχείο για το UNIX, θα έπρεπε τα προγράμματα να ανοίγουν το αρχείο που αντιπροσωπεύει

την οθόνη, να το γράφουν και να το ξανακλείνουν. Επειδή όμως η οθόνη αποτελεί την προκαθορισμένη έξοδο και είσοδο πολλών προγραμμάτων, το UNIX, θεωρεί ότι το αρχείο της οθόνης είναι πάντα ανοικτό και δεν ακολουθεί τις διαδικασίες (εντολές) ανοίγματος και κλεισίματος της οθόνης.

Επομένως όταν λέμε ότι έχουμε είσοδο-έξοδο δεδομένων από τον χρήστη εννοούμε ότι το πρόγραμμα διαβάζει από το αρχείο που αντιπροσωπεύει την οθόνη. Όταν έχουμε και είσοδο-έξοδο δεδομένων από/σε κάποιο αρχείο (file input-output) τότε θα πρέπει να γράψουμε ειδικές εντολές που θα ανοίγουν, διαβάζουν/ γράφουν και μετά θα κλείνουν το αρχείο.

Η προκαθορισμένη είσοδος δίνεται από: `$guess = <STDIN>;`

Η παραπάνω γραμμή θα εμφανιζόταν στο χρήστη σαν μια προτροπή ώστε να εισάγει τα δεδομένα του και ο,τιδήποτε εισάγει ο χρήστης θα αποθηκευτεί στη μεταβλητή `$guess`. Όπως έχουμε αναφέρει παραπάνω, ένα αρχείο αναγνωρίζεται εσωτερικά από το πρόγραμμα ως κάποιο αντικείμενο που ονομάζεται «file handler». Τα ονόματα που δίνουμε σε αυτά τα αντικείμενα είναι γραμμένα με κεφαλαία γράμματα. Τα «file handler» αποτελούν μία αναφορά/ παραπομπή σε κάποιο σενάριο ενός ανοιχτού αρχείου. Τα αρχεία που παριστούν την προκαθορισμένη είσοδο και έξοδο είναι πάντα ανοιχτά, έτσι ώστε να μη χρειάζεται να τα ανοίγουμε και να τα κλείνουμε σαν κανονικά αρχεία. Για παράδειγμα η εντολή `$stuff = <FILE_HANDLER>;` διαβάζει μία γραμμή εισόδου και την αποθηκεύει στη μεταβλητή `$stuff` από το «file handler» με όνομα FILE_HANDLER.

Η εντολή print είναι η προκαθορισμένη μέθοδος εξόδου δεδομένων. Για να εκτυπώσουμε την είσοδο/ τα δεδομένα στην οθόνη θα χρησιμοποιήσουμε:

```
print "$usersaid";
```

Με όλα τα παραπάνω μπορούμε να φτιάξουμε ένα μικρό παιχνίδι, το οποίο βρίσκει αν μαντέψαμε σωστά ένα αριθμό.

```
#!/usr/local/bin/perl
print "\nPlease enter a number: ";
$guess = <STDIN>;
while ($guess != 20) {
print "\nSorry! You guessed wrong! Please, guess again!";
$guess = <STDIN>;
}

print "Congratulations! You got it!";
```

Μπορούμε να έχουμε είσοδο ή έξοδο από αρχείο με μερικά βήματα ακόμα. Πρώτα ανοίγουμε το αρχείο:

```
open (NAMES, "/usr/people/ferm/names.txt");
```

Το "file handler" στην συγκεκριμένη περίπτωση είναι το NAMES.
Όταν έχουμε ανοίξει το αρχείο, μπορούμε να διαβάσουμε μία γραμμή του γράφοντας:

```
$fileinput = <NAMES>;
```

και να την αποθηκεύσουμε σε μία βαθμωτή μεταβλητή ή να χρησιμοποιήσουμε πίνακα όπου κάθε θέση είναι μία ξεχωριστή γραμμή:

```
@fileinput = <NAMES>;
```

Εφόσον έχουμε τα δεδομένα, πρέπει να βεβαιωθούμε ότι κλείσαμε το file handler στην περίπτωση που θέλουμε να γράψουμε πάλι κάτι σε αυτό.
***Σημείωση*:** Ορίζεται ρητά ότι δεν μπορούμε να διαβάζουμε και να γράφουμε ταυτόχρονα στο ίδιο αρχείο.

Κλείνουμε ένα αρχείο γράφοντας:

```
close (NAMES);
```

Υπάρχουν δύο διαφορετικοί μέθοδοι για να γράψουμε σε αρχείο: εγγραφή (writing) και προσάρτηση (appending). Κατά την εγγραφή διαγράφουμε το περιεχόμενο του αρχείου και ξαναγράφουμε το αρχείο από την αρχή. Αν το αρχείο δεν υπάρχει, δημιουργείται ένα νέο αρχείο. Κατά την προσάρτηση προσθέτουμε νέα πληροφορία στο τέλος του υπάρχοντος αρχείου.

Άνοιγμα αρχείου για εγγραφή:

```
Open (OUTFILE, ">/usr/people/perllog.txt");
```

Άνοιγμα αρχείου για προσάρτηση:

```
Open (OUTFILE, ">>/usr/people/perllog.txt");
```

Αφού αποφασίσουμε τον τρόπο εγγραφής, ας δούμε πώς μπορούμε να εκτυπώσουμε/ γράψουμε σε ένα αρχείο. Ακολουθείται η ίδια διαδικασία όπως κατά την εκτύπωση στην προκαθορισμένη έξοδο, με την εξαίρεση ότι καθορίζουμε ακριβώς που θέλουμε να εκτυπωθούν τα δεδομένα μας, θέτοντας το

«file handler» μεταξύ της εκτύπωσης και της πρότασης που θέλουμε να γραφεί στο αρχείο:

```
print OUTFILE "Here's a line of output";
```

### 5.4.7 Η συνάρτηση split

Η συνάρτηση Split ψάχνει για ένα δείγμα σε ένα αλφαριθμητικό και σπάει το αλφαριθμητικό σε έναν πίνακα βασισμένη στο δείγμα. Η χρήση της συνάρτησης γίνεται σύμφωνα με τον τύπο: split(/έκφραση_διαχωρισμού/, έκφραση_προς_διαχωρισμό).

Στο παρακάτω παράδειγμα η εντολή `@stuff = split (/ /, $stuff);` θα διασπάσει τα περιεχόμενα της μεταβλητής `$stuff`, χρησιμοποιώντας ως διαχωριστικό το κενό « ». Το αποτέλεσμα της split στο παρακάτω παράδειγμα είναι ότι ο πίνακας @stuff θα έχει τα εξής περιεχόμενα: @stuff=(“Learning”,”Perl”,”fast”).

```
#!/usr/local/bin/perl
# Αλφαριθμητική μεταβλητή (string).

$stuff = "Learning Perl fast";

# Διάσπαση του αλφαριθμητικού σε πίνακα με διαχωριστικό το
κενό.
@stuff = split (/ /, $stuff);

# Εκτύπωση κάθε θέσης του πίνακα σε διαφορετική γραμμή.
for ($i = 0; $i <= $#stuff; $i++) {
       print "Slot $i: $stuff[$i]\n";
}

# Εκτύπωση του αριθμού των λέξεων του αλφαριθμητικού.
print "There were $#stuff words.\n";
```

### 5.4.8 Μετατρέποντας την Perl σε CGI

Η συγγραφή σεναρίων CGI προϋποθέτει την γνώση κατασκευής φορμών σε HTML. Στο κεφάλαιο 3 μιλήσαμε για τις φόρμες εισαγωγής δεδομένων και την ετικέτα <FORM> της HTML. Υπάρχουν δύο μέθοδοι για να στείλουμε δεδομένα από το φυλλομετρητή μας στον αντίστοιχο εξυπηρετητή. Στην παράμετρο METHOD της ετικέτας <FORM> ορίζουμε την μέθοδο που θα

χρησιμοποιηθεί και η οποία μπορεί να είναι είτε η GET είτε η POST, όπως φαίνεται παρακάτω:

```
<FORM METHOD="POST" ACTION="/cgi-bin/myscript.pl">
```

Στην παράμετρο ACTION τοποθετούμε τη διαδρομή για το πρόγραμμα CGI που θα επεξεργαστεί τα δεδομένα που θα στείλουμε. Στην παράμετρο METHOD επιλέγουμε τη μέθοδο (POST ή GET). Θεωρείται ως προεπιλογή η GET. Αν επιλέξουμε να χρησιμοποιήσουμε την GET τότε όταν στείλουμε την φόρμα θα εμφανιστούν/ ενσωματωθούν τα δεδομένα μας στην διεύθυνση URL που χρησιμοποιούμε. Συγκεκριμένα, έστω ότι κάνουμε μία αναζήτηση στην ιστοσελίδα με διεύθυνση: http://leykada.physics.auth.gr/ όταν στείλουμε τα δεδομένα στη γραμμή που εμφανίζεται η διεύθυνση URL θα υπάρχει κάτι σαν το παρακάτω.

```
http://leykada.physics.auth.gr/cgi-bin/heal-linksearch/heal-linksearch.cgi?
SearchTerm1=network&Elsevier=NO&Springer=NO&Wilson=NO&MCB=
NO&
Kluwer=NO&IOP=NO&ACS=NO&Oxford=NO&Wiley=YES&ACM=NO
&          Blackwell=NO&Taylor=NO&Lippincott=NO&Boolean=AND&
SearchTerm2=&Search=Search
```

Αν όμως επιλέξουμε την POST τα δεδομένα θα παραμείνουν κρυμμένα για τον απλό χρήστη του φυλλομετρητή. Άλλη διαφορά ανάμεσα στις δύο αυτές μεθόδους αποτελεί το γεγονός ότι με την GET μπορούμε να στείλουμε δεδομένα μήκους 1024 χαρακτήρων και μόνο. Επίσης, αν σε μία ιστοσελίδα έχουμε χρησιμοποιήσει την POST, τότε η επιλογή του φυλλομετρητή μας «επιστροφή στην προηγούμενη σελίδα» δεν θα λειτουργήσει όπως θα περιμέναμε και αυτό διότι τα δεδομένα που είχαμε εισάγει θα έχουν χαθεί. Δεν ισχύει το ίδιο με τη μέθοδο GET.

Τα CGI επεξεργάζονται δεδομένα εισόδου με διαφορετικό τρόπο από την Perl. Μόλις το πρόγραμμα CGI επεξεργαστεί τα δεδομένα εισόδου, τα μετατρέπει σε μεταβλητές που χρησιμοποιούνται με τον ίδιο τρόπο από τα CGI και τα σενάρια Perl. Η είσοδος στα CGI μπορεί να ανακτηθεί με δύο τρόπους: "get" και "post". Αν είμαστε απόλυτα σίγουροι για τον τύπο της εισόδου που θα πάρουμε, τότε χρειάζεται να χρησιμοποιήσουμε μόνο ένα τύπο πρόσβασης εισόδου. Για να καλύψουμε όμως όλες τις πιθανές περιπτώσεις, γράφουμε το εξής:

```
if ($ENV{'REQUEST_METHOD'} eq "GET") {
```

```
            $in = $ENV{'QUERY_STRING'};
      } else {
            $in = <STDIN>;
   }
```

Αυτή η εντολή απευθύνει μία ερώτηση στον εξυπηρετητή, αν η μέθοδος (ο τρόπος με τον οποίο ο εξυπηρετητής παίρνει την πληροφορία) είναι η "GET". Τότε το πρόγραμμα θα διαβάσει την πληροφορία από το Query String και θα την καταχωρίσει σε μία μεταβλητή που ονομάζεται $in. Αλλιώς, θα την διαβάσει μέσω της προκαθορισμένης εισόδου όπως ακριβώς η Perl.

### 5.4.9 Ένα πρόγραμμα CGI

Θα γράψουμε τώρα ένα μικρό πρόγραμμα για την αναζήτηση τραγουδιών, που μπορούμε να προσθέσουμε στην προσωπική μας σελίδα.

Έχουμε μία φόρμα που δέχεται μόνο ένα είδος εισόδου: τον τίτλο του τραγουδιού. Το πεδίο της φόρμας που χρησιμοποιείται από το χρήστη για την εισαγωγή των δεδομένων, έχει όνομα: "title". Στο πεδίο αυτό της φόρμας μπορούμε να γράψουμε έναν τίτλο τραγουδιού, με κεφαλαία ή πεζά και να τον αναζητήσουμε.

Φτιάχνουμε την φόρμα με τον παρακάτω κώδικα HTML:

```
<form                action="/cgi-bin/το_cgi_programma_mas"
method="get">
<input type="text" length="20" name="title"><br>
<input type="submit" value="Τα τραγούδια που προτιμώ">
</form>
```

Για βάση δεδομένων μπορούμε να χρησιμοποιήσουμε ένα απλό αρχείο κειμένου, οργανωμένο έτσι ώστε να είναι εύκολα προσπελάσιμο από ένα μικρό πρόγραμμα. Για την ανάκτηση δεδομένων από ένα αρχείο, πρέπει να γνωρίζουμε πώς θα είναι δομημένα / μορφοποιημένα αυτά τα δεδομένα εσωτερικά στο αρχείο.

Έστω ότι το αρχείο δεδομένων ονομάζεται music.dat και έχουμε ορίσει μία σειρά από κανόνες μορφοποίησης για το αρχείο αυτό. Για να είναι λειτουργικό το σενάριο θα πρέπει το αρχείο να έχει αυστηρή δομή και να περιέχει πληροφορία στην παρακάτω μορφή:

```
---- τίτλος τραγουδιού με μικρά γράμματα ----
```

```
τίτλος

περιγραφή


```

Έτσι έχουμε, για παράδειγμα τα δεδομένα :

```
---- tuesday ----

Tuesday is gone

Το Tuesday is gone είναι ένα τραγούδι των Lynyrd Skynyrd.
Περιλαμβάνεται στον ομώνυμο δίσκο του συγκροτήματος.
κτλ…….

---- eagle ----

Fly like an eagle

Οι Steve Miller Band κυκλοφόρησαν το 1976 το δίσκο τους
"Fly like an eagle" που περιέχει και το ομώνυμο τραγούδι.
κτλ…….
```

**Επεξεργασία των δεδομένων εισόδου**

Για να ανακτήσουμε την πληροφορία που αντιστοιχεί στα δεδομένα εισόδου του χρήστη (που αντιστοιχεί στον τίτλο του τραγουδιού που θα εισάγει ο χρήστης στο πεδίο με το όνομα "title" της φόρμας), ανοίγουμε το αρχείο δεδομένων μας με τον τρόπο που θα ανοίγαμε κάθε άλλο αρχείο στην Perl. Στη συνέχεια διαβάζουμε και τοποθετούμε το περιεχόμενο σε μία μεταβλητή τύπου πίνακα και τέλος κλείνουμε το αρχείο:

```
open (DATA, "/usr/people/music.dat");
my @data = <DATA>;
Close (DATA);
```

Το πρόγραμμα τώρα έχει όλη την πληροφορία από το αρχείο δεδομένων και είναι πλέον έτοιμο να ψάξει για τη συγκεκριμένη πληροφορία που ταιριάζει στην είσοδο που έχει δώσει ο χρήστης. Πρέπει να έχουν δοθεί στο χρήστη σαφείς οδηγίες για το πώς πρέπει να γράψει τον τίτλο του τραγουδιού. Όλο το πρόγραμμα μέχρι εδώ είναι:

```
# Αυτή θα είναι πάντα η πρώτη γραμμή ενός Perl προγράμματος!
#!/usr/local/bin/perl

# Άνοιγμα της βάσης/αρχείου δεδομένων music.dat και αποθήκευσή # του/της στη μεταβλητή @data.
open (DATA, "/kapou_brisketai_kai_auto/music.dat");
my @data = <DATA>;
close (DATA);

# Με το παρακάτω γίνεται τοποθέτηση των δεδομένων εισόδου στη    # μεταβλητή $in, ανεξάρτητα από την μέθοδο εισαγωγής που έχει               # χρησιμοποιηθεί.

my $in;
if ($ENV{'REQUEST_METHOD'} eq "GET") {
 $in = $ENV{'QUERY_STRING'};
} else {
  $in = <STDIN>;
 }
```

Ας δούμε λίγο τα δεδομένα εισόδου του χρήστη και τις τροποποιήσεις που πρέπει να κάνουμε ώστε τα δεδομένα να περάσουν στο πρόγραμμα σωστά. Για παράδειγμα, θα πρέπει να βεβαιωθούμε ότι δεν θα έχουμε διάφορους παράξενους χαρακτήρες, οι οποίοι δημιουργούνται όταν τα δεδομένα εισόδου μεταφραστούν σε κώδικα URI. (URI είναι ο κώδικας στον οποίο μεταφράζονται οι μη αλφαριθμητικοί χαρακτήρες κατά την αποστολή μιας φόρμας και είναι η μέθοδος με την οποία ο φυλλομετρητής περνάει την πληροφορία στο CGI.). Έτσι, μπορούμε να γράψουμε:

```
$in =~ s/%(..)/pack("c",hex($1))/ge;
```

Το $s αποτελεί μία μεταβλητή της Perl που χρησιμοποιείται για την εύρεση και αντικατάσταση χαρακτήρων μέσα σε ένα σύνολο χαρακτήρων (string). Ο τελεστής ισοδυναμίας "=~" συνδέει το κείμενο όπου έχουν αντικατασταθεί κάποιοι χαρακτήρες με το αρχικό.

Η συνάρτηση "s/πρωτότυπο_κείμενο/ κείμενο_για_αντικατάσταση/ge" ψάχνει για κανονικές εκφράσεις ανάμεσα στην πρώτη και δεύτερη κάθετο ( / ) και μετά τις αντικαθιστά με οτιδήποτε υπάρχει ανάμεσα στη δεύτερη και την τρίτη κάθετο. Η εντολή pack πακετάρει το κείμενο και το μετατρέπει σε χαρακτήρες αναγνώσιμους για τον άνθρωπο. Το g, στο τέλος της συνάρτησης,

δηλώνει ότι η αντικατάσταση θα γίνει σε όλες τις περιπτώσεις. Το e δηλώνει ότι το νέο κείμενο θα πρέπει να ελεγχθεί αν αποτελεί έκφραση της Perl και τότε μόνο να γίνει η αντικατάσταση. Παρακάτω θα συναντήσουμε και το /i το οποίο δηλώνει ότι η αντικατάσταση/αναζήτηση θα γίνει λαμβάνοντας υπόψη τα κεφαλαία και πεζά γράμματα.

Επομένως, η παραπάνω εντολή αναζήτησης και αντικατάστασης ψάχνει να δει αν κάποιοι χαρακτήρες ξεκινάνε με το σύμβολο % και τους μετατρέπει σε χαρακτήρες αναγνώσιμους, χρησιμοποιώντας την δεκαεξαδική τους αναπαράσταση.

Αν ο τίτλος του τραγουδιού που θα εισάγει ο χρήστης περιέχει περισσότερες από μία λέξεις, η μεταβλητή $in θα περιέχει σύμβολα "+" στη θέση των κενών. Γι' αυτό το λόγο θα γράψουμε:

```
$in =~ s/\+/ /g;
```

Αυτή η εντολή αναζήτησης και αντικατάστασης της Perl ψάχνει και απαλείφει όλα τα σύμβολα "+" και τα αντικαθιστά με κενά.

Τέλος, θα χρειαστεί να μετακινήσουμε τα δεδομένα εκτός της $in και να τους δώσουμε ένα πιο περιγραφικό όνομα:

```
my %music = split (/=/, $in);
```

Αυτή η γραμμή κώδικα διασπά τα περιεχόμενα της $in χρησιμοποιώντας σαν οριοθέτη το "=" σε ένα συσχετιζόμενο πίνακα με όνομα `%music`. Το my υποδηλώνει ότι η μεταβλητή έχει περιορισμένη ισχύ, είναι, δηλαδή, τύπου private. Αφού το όνομα του πεδίου της φόρμας εισαγωγής δεδομένων είναι το "title", τα δεδομένα θα αναζητηθούν μέσα στο αρχείο δεδομένων στη θέση `$music{'title'}`.

**Αναζήτηση και εκτύπωση αποτελεσμάτων**

Η εκτύπωση/εμφάνιση αποτελεσμάτων σε μία ιστοσελίδα είναι ισοδύναμη με την εκτύπωση στην προκαθορισμένη έξοδο, με την εξαίρεση ότι πρέπει να χρησιμοποιήσουμε ετικέτες HTML. Πρέπει να ορίσουμε στον φυλλομετρητή το είδος των αποτελεσμάτων. Στο παράδειγμα μας τα αποτελέσματα θα είναι ένα αρχείο HTML. Έτσι, προσθέτουμε στην πρώτη δήλωση εκτύπωσης που θα σταλεί στο φυλλομετρητή του χρήστη το παρακάτω:

```
Content-Type: text/html\n\n
```

Εκτυπώνουμε, στην συνέχεια, το γενικό περιεχόμενο της σελίδας των αποτελεσμάτων μας ως εξής:

```
print <<END;

Content-Type: text/html\n\n

<html>
<body bgcolor="#000000" text="#ffffff">
<center>
<h2>Results</h2>
</center>
<p>

END
```

Προσέξτε ότι χρησιμοποιήσαμε HTML μέσα στον κώδικα CGI. Είναι σαν να γράφουμε μία κανονική σελίδα HTML σε τμήματα, τα οποία όμως περιέχονται σε εντολές εκτύπωσης.

Μέχρι το σημείο αυτό, έχουμε εκτυπώσει την επικεφαλίδα "Results", την πληροφορία του χρήστη σε μία μεταβλητή και το αρχείο δεδομένων αποθηκευμένο σε μία μεταβλητή πίνακα. Τώρα θα πρέπει να προσπελάσουμε τον πίνακα και να προσπαθήσουμε να ταιριάξουμε τους τίτλους. Αν κάποιος τίτλος ταιριάζει στο ζητούμενο, θα εκτυπώσουμε τη σχετική πληροφορία. Αν δεν ταιριάζει θα εκτυπώσουμε ένα μήνυμα λάθους.

Για την προσπέλαση ενός πίνακα μπορούμε να χρησιμοποιήσουμε έναν βρόχο `for` ή ένα βρόχο `foreach`. Σε αυτό το παράδειγμα, θα χρησιμοποιήσουμε τον βρόχο `for`.

```
for ($i = 0; $i <= $#data; $i++) {
}
```

Αυτό διατρέχει τις θέσεις στον `@data` μέχρι να μην υπάρχουν άλλες θέσεις και σταματάει. Πρέπει να ελέγξουμε αν ο τίτλος είναι αποθηκευμένος σε μία θέση:

```
if ($data[$i] =~ /-- $music{'title'} --/i) {
}
```

Αν υπάρχει ταίριασμα, θα χρησιμοποιήσουμε τον ακόλουθο κώδικα μέσα στο `if`:

```
print <<END;
<b><font size="+1">$data[$i + 1]</font></b><br>
$data[$i + 2]<br>
END
last;
```

Με τις παραπάνω εντολές προσπαθούμε να εντοπίσουμε αν ο τίτλος που έχει συμπληρώσει ο χρήστης περιέχεται μέσα στο αρχείο δεδομένων. Όπως έχει προαναφερθεί στο αρχείο δεδομένων ο τίτλος της εγγραφής του τραγουδιού περικλείεται σε παύλες (--) και καταλαμβάνει όλο το μήκος μιας γραμμής. Στην αμέσως επόμενη γραμμή του αρχείου περιέχεται ο αναλυτικός τίτλος του τραγουδιού. Τέλος, η περιγραφή του τραγουδιού περιέχεται στην επόμενη γραμμή. Εάν υπάρχει ταίριασμα, δηλαδή ισχύει η συνθήκη: `if ($data[$i] =~ /-- $music{'title'} --/i)`, τότε η μεταβλητή του βρόχου `$i` θα «δείχνει» τη γραμμή που περιέχεται ο τίτλος της εγγραφής του τραγουδιού, δηλαδή αυτή που περικλείεται με παύλες (--). Έστω ότι η τιμή της μεταβλητής `$i` είναι 32 τότε στην γραμμή `[$i + 1]`, δηλαδή την 33, θα βρίσκεται ο αναλυτικός τίτλος του τραγουδιού και στη `[$i + 2]` δηλαδή την 34, η περιγραφή του. Σημειώστε, ότι για να βρούμε τους αριθμούς των γραμμών χρησιμοποιήσαμε έναν μαθηματικό τελεστή/τελεστή μαθηματικών πράξεων και όχι έναν τελεστή εκχώρησης. Με αυτόν τον τρόπο, η τιμή της μεταβλητής `$i` δεν αυξάνεται, αλλά διατηρεί τη τιμή 32. Στη συνέχεια, μόλις εκτυπωθεί η πληροφορία, χρησιμοποιείται η εντολή `last` η οποία και ορίζει τον τερματισμό του βρόχου.

Για να αντιμετωπίσουμε την περίπτωση που ο χρήστης δώσει έναν τίτλο που δεν υπάρχει στη βάση δεδομένων, θα πρέπει να προσθέσουμε μία ακόμα εντολή. Μία δήλωση "`elseif`" στο παραπάνω τμήμα εντολών `if` δίνει μία λύση στο πρόβλημα, διότι θα λειτουργήσει μόνο όταν το πρόγραμμα είναι στην τελευταία του επανάληψη και δεν έχει βρει κάτι που να ταιριάζει.

```
if ($data[$i] =~ /-- $music{'title'} --/i) {
      print <<END;
      <b><font size="+1">$data[$i + 1]</font></b><br>
      $data[$i + 2]<br>

      END
      last;
} elsif ($i == $#data) {

}
```

Αν το $i ισούται με τον αριθμό των θέσεων στον @data και ο τίτλος δεν υπάρχει σε καμία θέση, τότε μπορούμε να χρησιμοποιήσουμε την παρακάτω εντολή:

```
Print "<b><I>Sorry! The lyrics are missing!</i></b>";
```

Όλος ο κώδικας φαίνεται παρακάτω:

```
for ($i = 0; $i <= $#data; $i++) {
        if ($data[$i] =~ /-- $music{'title'} --/i) {

              print <<END;
              <b><font        size="+1">$data[$i        +
     1]</font></b><br>
                     $data[$i + 2]<br>

                END
                last;
       } elsif ($i == $#data) {
          print   "<b><i>   Sorry!   The   lyrics   are
missing!</i></b>";
        }
  }
```

Ολοκληρώνουμε το πρόγραμμα γράφοντας:

```
Print <<END;
<center><hr size=1 width=50%></center>
</body>
</html>

END
```

**Ανακεφαλαίωση**

Είδαμε ένα απλό πρόγραμμα CGI, ας το ονομάσουμε my_first_cgi. Για να εκτελεστεί το πρόγραμμα CGI θα πρέπει να αποθηκευτεί στο φάκελο /cgi-bin του εξυπηρετητή Ιστού, στον οποίο διαθέτουμε τα κατάλληλα δικαιώματα.

Παρακάτω παρουσιάζεται ο κώδικας του παραδείγματος μας, ολοκληρωμένα.

```
#!/usr/local/bin/perl
```

```
#-------------------------------------------------------
---------

# Ο παρακάτω τρόπος θα διαβάσει τα δεδομένα ανεξάρτητα από
το αν # η μέθοδος είναι η "get" ή η "post," και θα τα
αποθηκεύσει στην # $in.
#-------------------------------------------------------
---------

my $in;
if ($ENV{'REQUEST_METHOD'} eq "GET") {
       $in = $ENV{'QUERY_STRING'};
} else {
       $in = <STDIN>;
}

#-------------------------------------------------------
---------
# Η πρώτη εντολή αντικαθιστά τον χαρακτήρα «+» με το κενό.
# η δεύτερη εντολή μετατρέπει τους χαρακτήρες που
περιέχονται    # στην URI σε μη-αλφαριθμητικούς
# Η επόμενη εντολή αποθηκεύει την είσοδο στον %music
#-------------------------------------------------------
---------

$in =~ s/\+/ /g;
$in =~ s/%(..)/pack("c",hex($1))/ge;
my %music = split (/=/, $in);

#-------------------------------------------------------
---------# Αυτή η εντολή διαβάζει/τοποθετεί την βάση
δεδομένων στον πίνακα
# @data και κλείνει το αρχείο.
#-------------------------------------------------------
---------
open (DATA, "../htdocs/music.dat");
my @data = <DATA>;
close (DATA);

#-------------------------------------------------------
---------
# Εκτυπώνονται τα γενικά του εγγράφου HTML
```

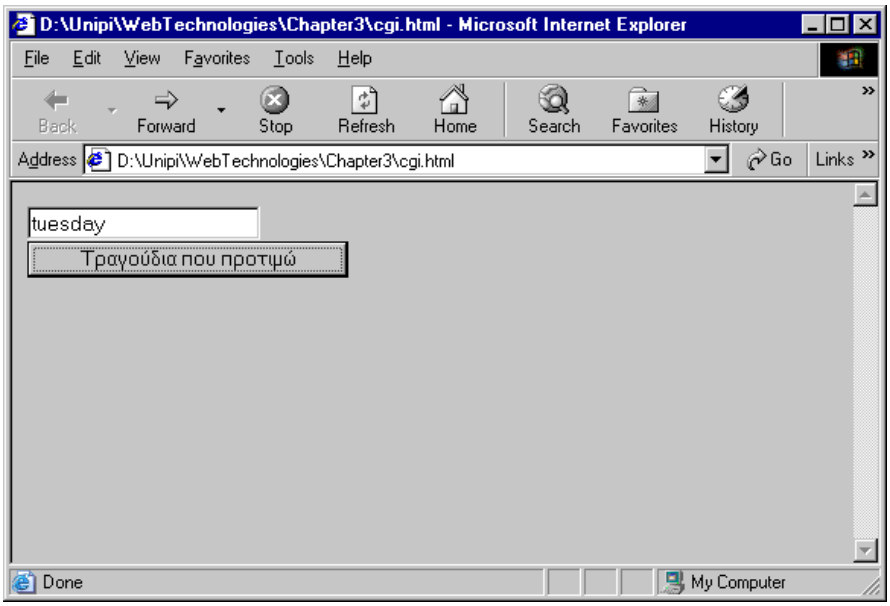
```
#-------------------------------------------------------
---------
print <<END;
Content-Type: text/html\n\n

<html>
<body bgcolor="#000000" text="#ffffff">
<center><h2>Results</h2></center><p>
<center><hr size=1 width=50%></center>
END
#-------------------------------------------------------
---------
# Ο παρακάτω βρόχος διατρέχει τον πίνακα ψάχνοντας και
# εκτυπώνοντας τον τίτλο, περιγραφή κτλ.
#-------------------------------------------------------
---------
for ($i = 0; $i <= $#data; $i++) {
       if ($data[$i] =~ /-- $music{'title'} --/i) {
              print <<END;
<center><b><font           size="+1">$data[$i           +
1]</font></b></center><br>
$data[$i + 2]<br>
END
              last;
       } elsif ($i == $#data) {
              print "<b><i>Sorry! I haven't reviewed that
one yet!</i></b>";
       }
}
#-------------------------------------------------------
---------# Εκτυπώνει τα τελικά για την ολοκλήρωση του
εγγράφου HTML.
#-------------------------------------------------------
---------
print <<END;
<center><hr size=1 width=50%></center>
</body>
</html>
END
```

Στο φάκελο /htdocs/, του εξυπηρετητή του Ιστού τοποθετούμε το αρχείο music.dat με τα δεδομένα μας, στην προκειμένη περίπτωση την συλλογή με τα τραγούδια, που έχει την δομή:

```
---- τίτλος τραγουδιού με μικρά γράμματα ----
τίτλος

περιγραφή
```

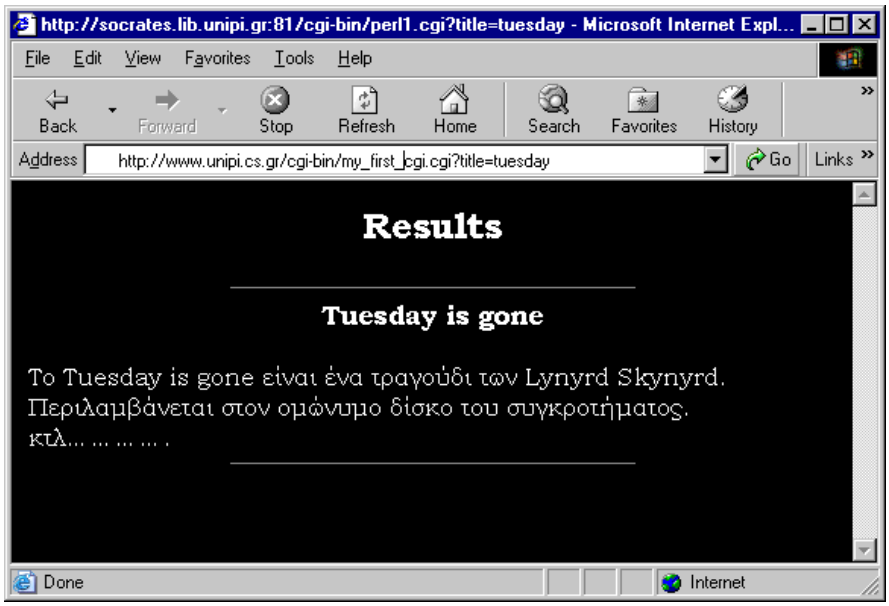
Έχουμε κατασκευάσει την φόρμα της Εικόνας 5.2.1, με την οποία θα ζητήσουμε από το πρόγραμμα CGI να αναζητήσει τα δεδομένα από το αρχείο `music.dat` και να επιστρέψει μια HTML σελίδα με τα αποτελέσματα.

```
<form                action="/cgi-bin/το_cgi_programma_mas"
method="get">
<input type="text" length="20" name="title"><br>
<input type="submit" value="Τραγούδια που προτιμώ">

</form>
```




**Εικόνα 5.8 – Φόρμα εισαγωγής δεδομένων**

Εάν ο χρήστης χρησιμοποιήσει την φόρμα της Εικόνας 5.2.1 για να αναζητήσει το τραγούδι με τίτλο «tuesday», τότε τα αποτελέσματα της αναζήτησης του θα εμφανιστούν όπως στην Εικόνα 5.2.2.




**Εικόνα 5.9 – Εμφάνιση αποτελεσμάτων**

Περισσότερες πληροφορίες για τη γλώσσα Perl μπορείτε να βρείτε στην διεύθυνση www.perl.com

## 5.4 Ερωτήσεις – Θέματα για ανάπτυξη– Ασκήσεις ReviewIT

| PHP | |
|---|---|
| Ποιο/ ποια από τα παρακάτω εισάγει κώδικα PHP ; | 1. <?<br>2. <%<br>3. και τα δύο<br>4. κανένα |
| Με ποιο/ποια γλώσσα προγραμματισμού από τις παρακάτω σχετίζεται η PHP; | 1. ASP<br>2. perl<br>3. HTML<br>4. XML |
| Τι σημαίνει PHP ; | 1. personal homepage tool<br>2. hypertext pre-processor<br>3. programming hypertext pages<br>4. personal<br>5. hypertext programming |
| Πώς μπορείτε να εμφανίσετε ένα κείμενο στο κάποιο έγγραφο; | 1. print '….';<br>2. echo '…';<br>3. και με τα δύο<br>4. με κανένα από τα δύο |
| Πως δημιουργείται μία array μεταβλητή ; | 1. var = array {'value1','value2'};<br>2. var = array ['value1','value2'];<br>3. var = array ('value1','value2');<br>4. var = array 'value1','value2'; |
| Με ποια βάση δεδομένων η PHP συνεργάζεται βέλτιστα; | 1. MS SQL<br>2. MySQL<br>3. MS Access<br>4. Oracle |
| Πώς συγκρίνουμε τις τιμές δύο μεταβλητών; | 1. $x == $y;<br>2. $x = $y;<br>3. $x eguals $y<br>4. $x is $y |
| Ποιο από τα παρακάτω δεν πραγματοποιείται με κώδικα PHP; | 1. Η διαγραφή αρχείων από τον εξυπηρετητή<br>2. Η ανακατεύθυνση σε άλλη ιστοσελίδα<br>3. Η αλλαγή τις τιμής στην status bar του φυλλομετρητή<br>4. Η ανάγνωση της διεύθυνσης IP του επισκέπτη σε μία ιστοσελίδα. |

| | |
|---|---|
| Με ποιο σύμβολο ορίζουμε μία μεταβλητή; | 1. %var<br>2. $var<br>3. &var<br>4. ?var |
| **PERL** | |
| Ποιο (ή ποια) αποτελεί σωστό όνομα για μία αριθμητική μεταβλητή; | 1. $hello<br>2. $_test<br>3. $trala_la_la_la_la_la_<br>4. $fries&gravy<br>5. $96tears<br>6. $tea_for_2 |
| Ποιες λειτουργίες έχουν αυτοί οι τελεστές; | 1. &&<br>2. &<br>3. ^<br>4. ne<br>5. . |
| Ποια η τιμή των πράξεων; | 1. 17 * 2 ** 3 / 9 % 2 << 2<br>2. 0 && (171567 * 98275 / 1174.5 ** 4)<br>3. 1171 ^ 904<br>4. "abc" . "de" x 2 |
| Υποθέστε ότι έχουν γίνει οι παρακάτω αναθέσεις τιμών :<br>@list = (1, 2, 3);<br>$scalar1 = "hello";<br>$scalar2 = "there";<br>Τι είναι αποθηκευμένο στη μεταβλητή @newlist σε καθεμία περίπτωση: | 1. @newlist = @list;<br>2. @newlist = reverse(@list[1,2]);<br>3. @newlist = ($scalar1, @list[1,1]);<br>4. ($dummy, @newlist) = @list;<br>5. @newlist[2,1,3] = @list[1,2,1];<br>6. @newlist = <STDIN>; |
| Μπορώ να γράψω σε ένα αρχείο και να το διαβάσω αργότερα; | Ναι αλλά όχι ταυτόχρονα. Για να διαβάσεις ένα αρχείο στο οποίο έχει γράψει κάτι θα πρέπει να το κλείσεις πρώτα και μετά να το ξανανοίξεις για διάβασμα. |
| Πόσες φορές επαναλαμβάνεται ο κάθε βρόχος; | 1. for ($count = 0; $count < 7; $count++) { print ("$count\n"); }<br>2. $count = 1;<br>do { |

| | |
|---|---|
| | print ("$count\n");<br>} until ($count++ > 10);<br>3. for ($count = 1; $count <= 10; $count++) {<br>last if ($count == 5);<br>} |

**Θέματα για Ανάπτυξη**

1. Πότε τερματίζεται ένας βρόχος while;
2. Πότε μία δήλωση until τερματίζει έναν βρόχο;
3. Ποια η διαφορά ανάμεσα στους τελεστές ‘=’, ‘==’, ‘===’ ;
4. Ποιο είναι το αποτέλεσμα της παρακάτω δήλωσης:14 + 6 * 3 - 10 / 2
5. Ποιο το αποτέλεσμα του παρακάτω κώδικα print_r ($hight_school);, της ενότητας «Πίνακες»;
6. Τι σημαίνουν τα $a++ και ++$a. Ποια η διαφορά τους;

Ασκήσεις

1. Φτιάξτε ένα πρόγραμμα σε PERL που διαβάζει δύο αριθμούς από μία φόρμα HTML, τους προσθέτει και εκτυπώνει το αποτέλεσμά τους.
2. Γράψτε ένα πρόγραμμα σε PERL που διαβάζει δύο αριθμούς από μία φόρμα HTML, διαιρεί τον πρώτο με τον δεύτερο και εκτυπώνει το αποτέλεσμα, αλλά:
   Εκτυπώνει μήνυμα λάθους εάν ο δεύτερος αριθμός είναι 0.
   Εάν ο πρώτος αριθμός είναι 0 και ο δεύτερος 1 τότε απλά εκτυπώνει τον πρώτο, αφού δεν χρειάζεται να κάνει κάποια διαίρεση.
3. Γράψτε ένα πρόγραμμα που χρησιμοποιεί τον βρόχο while και εκτυπώνει τους αριθμούς από το 1 έως το 10 σε φθίνουσα ταξινόμηση.
4. Γράψτε ένα πρόγραμμα που χρησιμοποιεί τον βρόχο until για να εκτυπώσει τους αριθμούς από το 1 έως το 10 σε φθίνουσα ταξινόμηση.
5. Γράψτε ένα πρόγραμμα που διαβάζει δύο αριθμούς από μία φόρμα HTML, διαιρεί τον πρώτο με τον δεύτερο και εκτυπώνει το αποτέλεσμα, και το υπόλοιπο.
6. Γράψτε ένα πρόγραμμα που μετράει όλες τις εμφανίσεις της λέξης, που δίνει ο χρήστης από μία φόρμα HTML, σε ένα αρχείο δεδομένων.
7. Γράψτε τα προγράμματα των ασκήσεων 1-6 χρησιμοποιώντας PHP.