Υπερ-καθολικές μεταβλητές (Superglobals)

Οι μεταβλητές που μπορούμε να ορίσουμε στην PHP μπορεί να έχουν καθολική ή τοπική εμβέλεια. Εκτός όμως από τις δικές μας μεταβλητές η PHP παρέχει ένα έτοιμο σύνολο υπερ-καθολικών μεταβλητών (superglobals), που δεν χρειάζεται να τις αρχικοποιήσουμε εμείς, με τις οποίες διατίθεται στον προγραμματιστή ένα σύνολο πληροφοριών που σχετίζονται με το κάθε αίτημα (request) ή μια συνεδρία (session) του χρήστη.

Οι superglobals είναι:

- $GLOBAL
- $_SERVER
- $_REQUEST
- $_POST
- $_GET
- $_FILES
- $_ENV
- $_COOKIE
- $_SESSION

Οι μεταβλητές είναι όλες πίνακες. Δεν θα δούμε αναλυτικά όλες τους αλλά θα επικεντρωθούμε μόνο στις, **$_SERVER**, **$_POST**, **$_GET** και **$_SESSION**. Αναλυτικές πληροφορίες για τις μεταβλητές αυτές μπορείτε να βρείτε με μια αναζήτηση στο διαδίκτυο, όπως για παράδειγμα στη σελίδα του w3school (http://www.w3schools.com/php/php_superglobals.asp).

$_SERVER

Η μεταβλητή αυτή περιέχει πληροφορίες για το τρέχον αίτημα που εξυπηρετείται. Φυλάει πληροφορίες τόσο για τον server και το script που εκτελείται όσο και πληροφορίες για τον χρήστη που το ζήτησε. Φυσικά, όπως όλες οι superglobals είναι πίνακας και έχει προκαθορισμένους δείκτες που μπορούμε να χρησιμοποιήσουμε. Ακολουθούν κάποιοι ενδεικτικοί δείκτες:

$_SERVER['PHP_SELF']	Επιστρέφει το όνομα του αρχείου από το script που εκτελείται
$_SERVER['SERVER_ADDR']	Επιστρέφει την διεύθυνση IP του Server
$_SERVER['SERVER_NAME']	Επιστρέφει το όνομα του server (πχ sch.gr)
$_SERVER['REQUEST_METHOD']	Επιστρέφει το τύπο του αιτήματος που έγινε (για παράδειγμα POST)
$_SERVER['REQUEST_TIME']	Επιστρέφει το χρόνο έναρξης εκτέλεσης του request (με τη μορφή ενός timestamp)
$_SERVER['REMOTE_ADDR']	Επιστρέφει την IP του χρήστη που επισκέπτεται τη σελίδα

$_SESSION

Η μεταβλητή αυτή μπορεί να χρησιμοποιηθεί από τον προγραμματιστή για να αποθηκεύει πληροφορίες που θα είναι διαθέσιμες καθ' όλη τη διάρκεια που ένας χρήστης είναι συνδεδεμένος με τη σελίδα. Κανονικά ένα request ολοκληρώνει το κύκλο του με την εκτέλεσή του και την αποστολή του τελικού HTML στο χρήστη. Όλες οι πληροφορίες που αποθηκεύονταν σε μεταβλητές της PHP χάνονται, εκτός και αν αποθηκεύονται στη βάση. Με τη χρήση της μεταβλητής $_SESSION μπορούμε να διατηρήσουμε κάποιες πληροφορίες στη μνήμη του server όσο ένας χρήστης παραμένει ενεργός σε μια σελίδα, χωρίς να χρειαστεί να τις αποθηκεύσουμε στη βάση (εκτός και αν το επιθυμούμε). Με αυτό τον τρόπο διατηρούμε πληροφορίες που διευκολύνουν τη πλοήγηση του χρήστη, επιτρέποντας τη μεταφορά/διατήρηση πληροφοριών από το ένα request στο επόμενο. Όταν ο χρήστης κλείσει τον browser τότε η συνεδρία του χρήστη με τον server ολοκληρώνεται και η μεταβλητή $_SESSION καταστρέφεται.

Ας θεωρήσουμε ότι επιθυμούμε να μετρήσουμε πόσες σελίδες από τον ιστο-χώρο μας έχει επισκεφτεί ένας χρήστης. Αυτό σημαίνει ότι πρέπει να διατηρούμε μια μεταβλητή η οποία δεν θα χάνεται κατά τη πλοήγηση του χρήστη και θα αυξάνεται κατά ένα με κάθε καινούργιο request (επίσκεψη μιας σελίδας). Μπορούμε λοιπόν να βάλουμε σε κάθε σελίδα (ή σε ένα script που χρησιμοποιείται από όλες τις σελίδες, όπως είναι συνήθως η επικεφαλίδα header) ένα μικρό κομμάτι κώδικα που θα ενημερώνει τη μεταβλητή $_SESSION. Απαραίτητο είναι να καλείται πρώτα η συνάρτηση **session_start()**, που επιτρέπει τη πρόσβαση σε αυτή τη μεταβλητή.

```
<?php
	session_start();
	// ελέγχουμε αν υπάρχει ο δείκτης counter στο SESSION
	// βοηθά στην αρχικοποίηση του counter κατά το πρώτο request
	if (!isset($_SESSION['counter']))
		$_SESSION['counter'] = 0;

	$_SESSION['counter']++;
	echo $_SESSION['counter'];
?>
```

Με τη χρήση της $_SESSION μπορούμε να φυλάξουμε διάφορες πληροφορίες που διευκολύνουν τη πλοήγηση του χρήστη στις σελίδες μας, όπως για παράδειγμα τη γλώσσα που έχει επιλέξει για τη πλοήγηση, τις πληροφορίες σύνδεσης (εφόσον απαιτείται Login) ή ακόμα το νόμισμα και το καλάθι όταν επισκεπτόμαστε ένα ηλεκτρονικό κατάστημα.

$_GET και $_POST

Οι δυο αυτές μεταβλητές χρησιμοποιούνται κατά την αποστολή πληροφοριών από τον χρήστη προς τον server. Βασική διαφορά τους είναι ότι η **$_GET** χρησιμοποιείται όταν στέλνουμε πληροφορίες μέσω του **URL**, δηλαδή τις ενσωματώνουμε στον σύνδεσμο του script που μας ενδιαφέρει. Αντίθετα οι πληροφορίες που στέλνονται με τη **$_POST** δεν είναι τόσο φανερές, συνοδεύουν το request προς τον Server και επιτρέπουν μεγαλύτερη ασφάλεια και περισσότερα δεδομένα προς αποστολή.

Η **$_POST** συνήθως χρησιμοποιείται για την αποστολή των στοιχείων μιας **HTML** φόρμας. Αυτή τη δουλειά μπορεί να τη κάνει και η **$_GET** αλλά συνήθως αποφεύγεται. Αντίθετα η **$_GET** χρησιμοποιείται για ενσωματώσουμε στο request κριτήρια ή διευκρινιστικές πληροφορίες. Για παράδειγμα το URL ενός video στο YouTube έχει τη μορφή

https://www.youtube.com/watch?v=UObINRj2EGY

Το ερωτηματικό «?» διαχωρίζει το μονοπάτι του script που θέλουμε να εκτελέσουμε από τις παραμέτρους που θέλουμε να συνοδεύουν τη κλήση μας. Έτσι μπορούμε να πούμε ότι το τμήμα «www.youtube.com/watch» ενημερώνει την ιστοσελίδα ότι επιθυμούμε να παρακολουθήσουμε ένα video και το τμήμα «?v=UObINRj2EGY» παρέχει το κωδικό του video που θέλουμε.

Όταν η PHP δέχεται ένα τέτοιο αίτημα, εντοπίζει τις παραμέτρους που βρίσκονται μετά το ερωτηματικό και τις αποθηκεύει στη superglobal **$_GET**, από όπου είναι διαθέσιμες στον προγραμματιστή της σελίδας. Στη προκειμένη περίπτωση είναι μόνο μια παράμετρος αλλά αν θέλαμε θα μπορούσαμε να έχουμε περισσότερες χωρισμένες με «&». Για παράδειγμα μια κλήση της μορφής

www.mydomain.com/search.php?text=search+text&sort=ID&order=ASC

Σε αυτή την εικονική κλήση καλούμε ένα script που βρίσκεται στο αρχείο search.php και ως παραμέτρους περνάμε τις text, sort και order. Η PHP θα πάρει αυτές τις παραμέτρους και θα τις αποθηκεύσει στη **$_GET**. Έτσι στον προγραμματιστή θα είναι διαθέσιμος ένας πίνακας με τις εξής τιμές

$_GET[‘text’] = “search text”

$_GET[‘sort] = “ID”

$_GET[‘order’] = “ASC”

Φυσικά το τι θα κάνει ο προγραμματιστής στο script με αυτές τις παραμέτρους, αφορά τον προγραμματιστή και την επιθυμητή λειτουργία της σελίδας. Επίσης ο χρήστης δεν έχει να ασχοληθεί με τη μορφή του URL και τις παραμέτρους που θα μπουν εκεί. Τα URL’s κατασκευάζονται από την PHP και ενσωματώνονται στη σελίδα με τη μορφή συνδέσμων, που ο χρήστης ενεργοποιεί με το click.

Παράδειγμα με τη $_GET

Ας θεωρήσουμε ότι έχουμε ένα πίνακα με το ονοματεπώνυμο κάποιων μαθητών. Το κλειδί (Index) κάθε στοιχείου του πίνακα είναι το μοναδικό ID του κάθε μαθητή. Ο πίνακάς μας θα μπορούσε να είναι ο εξής:

3	Γιώργος Αναστασίου
6	Νίκος Γεωργίου
8	Παναγιώτα Πετροπούλου
9	Αναστασία Παπαδοπούλου
12	Αντώνης Βασιλείου

Θέλουμε να φτιάξουμε μια σελίδα (έστω **list.php**) που θα εμφανίζει αυτά τα στοιχεία αλλά να δίνεται στο χρήστη η επιλογή να τα ταξινομήσει ως προς το ID ή ως προς το όνομα, με αύξουσα ή φθίνουσα σειρά. Έτσι στο επάνω μέρος θα βάλουμε τέσσερις επιλογές:

Ταξινόμηση: ID αύξουσα / ID φθίνουσα / Όνομα αύξουσα / Όνομα φθίνουσα

Φυσικά η λύση που προτείνεται είναι απλοϊκή αλλά βολεύει για τη κατανόηση της χρήσης της **$_GET**. Κάθε μια από αυτές τις επιλογές θα είναι ένα Link που θα οδηγεί στο αρχείο list.php με διαφορετικές παραμέτρους κάθε φορά. Τα ονόματα και οι πιθανές τιμές των παραμέτρων στα URL μπορεί να είναι οτιδήποτε, αρκεί στην php να χρησιμοποιήσουμε τα ίδια. Έτσι τα links για αυτές τις επιλογές θα μπορούσαν να είναι σε **HTML**

```
<a href="list.php?sort=id&order=ASC"> ID αύξουσα</a>
<a href="list.php?sort=id&order=DESC"> ID φθίνουσα </a>
<a href="list.php?sort=name&order=ASC"> Όνομα αύξουσα </a>
<a href="list.php?sort=name&order=DESC"> Όνομα φθίνουσα </a>
```

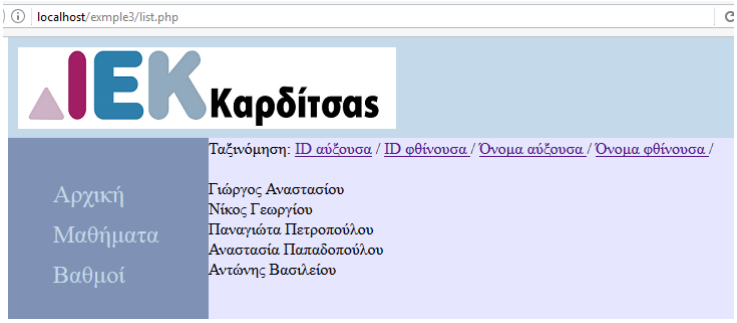
Εδώ κάναμε τη σύμβαση ότι η παράμετρος sort θα δηλώνει ως προς ποια ιδιότητα θα ταξινομούμε και η παράμετρος order αν η ταξινόμηση θα είναι αύξουσα (**ASC**) ή φθίνουσα (**DESC**). Αυτούς τους πιθανούς συνδυασμούς παραμέτρων θα πρέπει να περιμένουμε και στο PHP script ώστε να δείξουμε τα αποτελέσματα με τον κατάλληλο τρόπο.

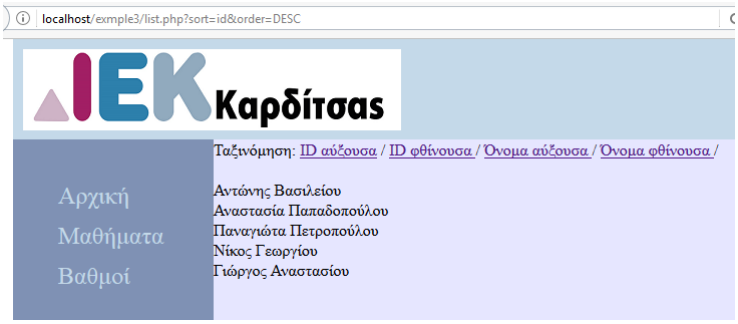
```
Ταξινόμηση:
<a href='list.php?sort=id&order=ASC'> ID αύξουσα</a> /
<a href='list.php?sort=id&order=DESC'> ID φθίνουσα </a> /
<a href='list.php?sort=name&order=ASC'> Όνομα αύξουσα </a> /
<a href='list.php?sort=name&order=DESC'> Όνομα φθίνουσα </a> /
<br><br>

<?php

	$math = array(
		3 => 'Γιώργος Αναστασίου',
		6 => 'Νίκος Γεωργίου',
		8 => 'Παναγιώτα Πετροπούλου',
		9 => 'Αναστασία Παπαδοπούλου',
		12 =>'Αντώνης Βασιλείου',
	);
	// πρώτα ελέγχουμε αν υπάρχουν οι παράμετροι
	// αν υπάρχουν θα ταξινομήσουμε τον πίνακα κατάλληλα
	if (isset($_GET['order']) && isset($_GET['sort'])) {
		if ($_GET['sort'] == 'id' && $_GET['order']=='ASC')
			ksort($math);
		if ($_GET['sort'] == 'id' && $_GET['order']=='DESC')
			krsort($math);
		if ($_GET['sort'] == 'name' && $_GET['order']=='ASC')
			asort($math);
		if ($_GET['sort'] == 'name' && $_GET['order']== 'DESC')
			arsort($math);
	}
	foreach ($math as $mathitis) {
		echo $mathitis.'<br>';
	}
?>
```

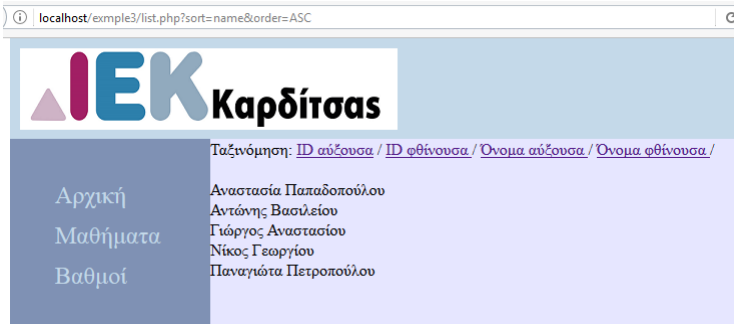
Οι συναρτήσεις ταξινόμησης που χρησιμοποιήθηκαν είναι οι εξής:

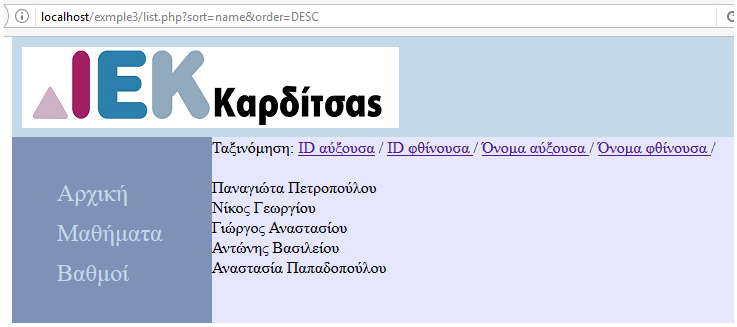
- **asort**(array) – Ταξινομεί τα στοιχεία του πίνακα σε **αύξουσα** τάξη μεγέθους. Χρησιμοποιείται κυρίως σε **associative** πίνακες.
- **arsort**(array) – Ταξινομεί τα στοιχεία του πίνακα σε **φθίνουσα** τάξη μεγέθους. Χρησιμοποιείται κυρίως σε **associative** πίνακες.
- **ksort**(array) – Ταξινομεί τα στοιχεία του πίνακα σε **αύξουσα** τάξη μεγέθους του κλειδιού και όχι του στοιχείου του ίδιου.
- **krsort**(array) – Ταξινομεί τα στοιχεία του πίνακα σε **φθίνουσα** τάξη μεγέθους του κλειδιού και όχι του στοιχείου του ίδιου.

Ακολουθούν εικόνες με από την όψη της ιστοσελίδας ανάλογα με το link που έχει χρησιμοποιηθεί, το οποίο φαίνεται στο URL στη γραμμή διεύθυνσης (με το σκελετό που χρησιμοποιείται στα παραδείγματα του εργαστηρίου).









Γενικά μπορούμε να πούμε ότι η δομή του αρχείου php για τη δημιουργία και λειτουργία της σελίδας που χρησιμοποιεί τη $_GET, περιλαμβάνει:

- Έλεγχο των παραμέτρων/πληροφοριών που στέλνει ο χρήστης
- Προσαρμογή των δεδομένων εξόδου στις παραμέτρους του χρήστη
- Δημιουργία των links με τις απαραίτητες παραμέτρους
- Δημιουργία του υπόλοιπου περιεχομένου της σελίδας

Παράδειγμα με τη $_POST

Όπως είπαμε η μεταβλητή **$_POST** χρησιμοποιείται συνήθως όταν ο χρήστης συμπληρώνει και στέλνει στον **server** μια **φόρμα δεδομένων**. Ας υποθέσουμε ότι θέλουμε να φτιάξουμε μια σελίδα με το όνομα **form.php**, με την οποία ο χρήστης θα μπορεί να συμπληρώσει το όνομα και το βαθμό ενός μαθήματος και να τα υποβάλει στο Server για αποθήκευση. Η δομή του αρχείου θα είναι λίγο διαφορετική από αυτή του προηγούμενου παραδείγματος

- εφόσον υπάρχουν πληροφορίες από τον χρήστη
  - Έλεγχος των πληροφοριών (αν το επιθυμούμε)
  - επεξεργασία των πληροφοριών
  - μεταφορά στη κατάλληλη σελίδα μετά την επιτυχή αποθήκευση
- Δημιουργία της φόρμας και του υπόλοιπου περιεχομένου της σελίδας

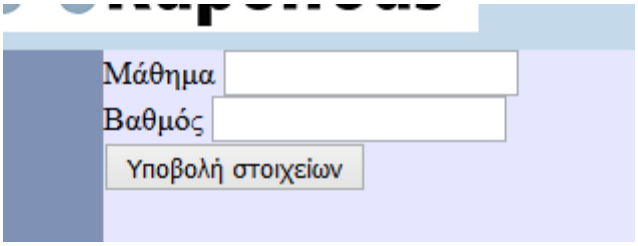
Ας δούμε πρώτα το κώδικα HTML για τη δημιουργία της φόρμας. Όλα τα πεδία που χρειαζόμαστε μπαίνουν σε ένα tag **form**, στο οποίο πρέπει να δηλώσουμε ποιο php script θα δεχτεί τα δεδομένων κατά την υποβολή και με ποιο τρόπο θα γίνει η υποβολή. Αυτά τα στοιχεία δηλώνονται με τις ιδιότητες **action** και **method** στη δήλωση της φόρμας.

```
<form action=" form.php" method="post">
…
</form>
```

Εδώ βλέπουμε ότι το ίδιο αρχείο που θα δημιουργήσει τη φόρμα θα είναι και αυτό που θα υποδεχθεί τα στοιχεία κατά την υποβολή τους. Επίσης η μέθοδος αποστολής θα είναι η POST. Μέσα στη φόρμα θα πρέπει να βάλουμε τα πεδία που θέλουμε με τη χρήση του tag **input**. Σημαντικό σε αυτά τα tags είναι να ορίσουμε την ιδιότητα **name**, με την οποία ορίζουμε πως θα λέγεται η μεταβλητή που θα κουβαλήσει το περιεχόμενο στη PHP. Για την υποβολή της φόρμας προσθέτουμε ένα κουμπί με τη χρήση ενός ειδικού **input** πεδίου που έχει τύπο **submit**.

Ακολουθεί ο κώδικας και η μορφή της φόρμας που θα εμφανιστεί

```
<form action=" form.php" method="post">
      <label>Μάθημα</label>
      <input name="mathima" type="text">
      <br>
      <label>Βαθμός</label>
      <input name="vathmos" type="text">
      <br>
      <input type="submit" value="Υποβολή στοιχείων">
</form>
```



Η φόρμα που δημιουργείται μπορεί να δεχτεί πλέον στοιχεία και να τα υποβάλει στο script **form.php** με το πάτημα του κουμπιού. Η υποβολή όμως αυτή δεν θα έχει κανένα αποτέλεσμα αφού στο **script** δεν υπάρχει κατάλληλος κώδικας για την υποδοχή και επεξεργασία των στοιχείων. Για να δούμε και να επεξεργαστούμε τα στοιχεία στη PHP θα χρησιμοποιήσουμε τη μεταβλητή **$_POST** στη οποία αποθηκεύονται αυτόματα τα δεδομένα της φόρμας, όταν γίνεται υποβολή.

Προσοχή: Οι τιμές της φόρμας αποθηκεύονται σε θέσεις του πίνακα **$_POST** και ως κλειδιά της κάθε θέσης χρησιμοποιείται το όνομα που χρησιμοποιήθηκε στο **input** πεδίο της φόρμας. Έτσι η τιμή που θα εισαχθεί στο πεδίο «<input name="vathmos" type="text">», θα είναι στη PHP διαθέσιμο ως **$_POST['vathmos']**.

Ας δούμε τώρα το κώδικα που θα επεξεργάζεται τις τιμές που υποβάλλονται (όταν υποβάλλονται, γιατί κατά την πρώτη επίσκεψη στη σελίδα δεν έχουν σταλεί στοιχεία)

```
<?php

	function save_in_database($mathima, $vathmos) {
		//κώδικας για την αποθήκευση στη βάση
	}
?>
<?php
	if ($_SERVER['REQUEST_METHOD'] == 'POST') {
		save_in_database($_POST['mathima'], $_POST['vathmos']);
	}
?>
<form action="form.php" method="post">
	<label>Μάθημα</label>
	<input name="mathima" type="text">
	<br>
	<label>Βαθμός</label>
	<input name="vathmos" type="text">
	<br>
	<input type="submit" value="Υποβολή στοιχείων">
</form>
```

Ο έλεγχος **if ($_SERVER['REQUEST_METHOD'] == 'POST')** χρησιμοποιεί τη μεταβλητή $_SERVER για να αποφασίσει αν έχει γίνει αποστολή στοιχείων από τη φόρμα. Αν αυτό είναι αληθές τότε τα στοιχεία αποθηκεύονται στη βάση με τη κλήση μιας συνάρτησης της οποίας το περιεχόμενο δεν μας ενδιαφέρει αυτή τη στιγμή. Με τη σύνδεση στη βάση δεδομένων θα ασχοληθούμε στο επόμενο κεφάλαιο.

Έλεγχος της φόρμας

Τις περισσότερες φορές θέλουμε να βάλουμε κάποιους περιορισμούς στα δεδομένα που θα εισάγει ο χρήστης, πριν τα αποθηκεύσουμε στη βάση ή γενικά τα επεξεργαστούμε. Οι περιορισμοί εξαρτώνται αποκλειστικά από εμάς και δεν υπάρχει κάποιος γενικός κανόνας. Για παράδειγμα στο προηγούμενο παράδειγμα δε θα θέλαμε το

όνομα του μαθήματος να είναι λιγότερο από τρεις χαρακτήρες και η βαθμολογία εκτός του συνόλου 1-10.

Ο έλεγχος για την ορθότητα των δεδομένων πρέπει να γίνει στο php script και αν τα δεδομένα δεν είναι αποδεκτά τότε η φόρμα να εμφανιστεί με τις πρόσφατες τιμές που έβαλε ο χρήστης με την επισήμανση ότι σε κάποια πεδία υπάρχει λάθος. Αυτό μπορεί να γίνει με πάρα πολλούς διαφορετικούς τρόπους και όχι μόνο με αυτόν που θα περιγράψουμε ενδεικτικά.

```
<?php

	function save_in_database($mathima, $vathmos) {
		//κώδικας για την αποθήκευση στη βάση
	}
?>
<?php
	$errors = array();
	$vathmos = '';
	$mathima = '';
	if ($_SERVER['REQUEST_METHOD'] == 'POST') {
		// ελέγχουμε κάθε πληροφορία χωριστά
		// αν εντοπίσουμε λάθος σε κάποια προσθέτουμε μια
		// καταχώριση στο πίνακα των λαθών

		if (strlen(($_POST['mathima']) < 3)
			$errors['mathima'] = true;

		if ($_POST['vathmos'] < 1 || $_POST['vathmos'] > 10)
			$errors['vathmos'] = true;

		if (empty($errors))
		save_in_database($_POST['mathima'], $_POST['vathmos']);
		else {
			echo 'Υπάρχουν λάθη στη φόρμα';
			$mathima = $_POST['mathima'];
			$vathmos = $_POST['vathmos'];
		}
	}
?>
<form action="form.php" method="post">
	<label>Μάθημα</label>
		<input name="mathima" value="<?php echo $mathima; ?>"
		<?php if (isset($errors['mathima'])) ?>
			echo 'style="border:2px solid red">';
		<?php } ?>
		type="text">
	<br>
	<label>Βαθμός</label>
		<input name="vathmos" value="<?php echo $vathmos; ?>"
		<?php if (isset($errors['vathmos'])) ?>
			echo 'style="border:2px solid red">';
		<?php } ?>
		type="text">
	<br>
	<input type="submit" value="Υποβολή στοιχείων">
</form>
```

Βλέπουμε στο παραπάνω παράδειγμα ότι χρησιμοποιείται ένας πίνακας $erros ο οποίος αρχικά είναι κενός. Ελέγχουμε μια μια τις τιμές της φόρμας για τυχόν λάθη και αν

βρούμε κάποιο δημιουργούμε μια καινούργια καταχώριση στο πίνακα των λαθών. Αν μετά τον έλεγχο ο πίνακας είναι ακόμα άδειος, αυτό σημαίνει ότι όλα τα δεδομένα είναι αποδεκτά και προχωρούμε στην επεξεργασία/αποθήκευση των δεδομένων. Αλλιώς τυπώνουμε κατάλληλο μήνυμα.

Ανάλογες τροποποιήσεις βλέπουμε και στην ετοιμασία της φόρμας. Σε περίπτωση λάθους σε κάποιο πεδίο όλα τα πεδία συμπληρώνονται με τις παλιές τους τιμές και βάζουμε κόκκινο περίγραμμα μόνο σε εκείνα τα πεδία που είχαν λάθος. Τονίζουμε ότι είναι ενδεικτική μέθοδος. Το πώς θα εντοπίσει και θα υποδείξει το λάθος ο κάθε προγραμματιστής μπορεί να διαφέρει.