ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΔΙΑΔΙΚΤΥΟΥ

Η ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ PHP
(TUTORIAL)

https://www.w3schools.com/php/DEFAULT.asp

ΘΕΣΣΑΛΟΝΙΚΗ

Χ. Γεωργιάδης

PHP - Tutorial

Η PHP είναι μια server scripting γλώσσα (γλώσσα για ανάπτυξη προγραμμάτων που εκτελούνται σε κάποιο server) και είναι ένα ισχυρό εργαλείο για τη δημιουργία δυναμικών και διαδραστικών ιστοσελίδων γρήγορα.

Η PHP είναι μια ευρέως χρησιμοποιούμενη, δωρεάν και αποτελεσματική εναλλακτική λύση για τους ανταγωνιστές της, όπως η ASP της Microsoft.

Παράδειγμα php_1.php

```
<!DOCTYPE html>
<html>
<body>

<?php
echo "My first PHP script!";
?>

</body>
</html>
```

Έξοδος: My first PHP script!

Στη πραγματικότητα, μετά την επεξεργασία του PHP κώδικα, ο server στέλνει το παρακάτω output στον browser:

```
<!DOCTYPE html>
<html>
<body>
   My first PHP script!!
</body>
</html>
```

Για να έχουμε υποστήριξη Ελληνικών χαρακτήρων, προσθέτουμε στο τμήμα <head> ετικέτα <meta>, όπως:

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

Εισαγωγή

Τα PHP scripts εκτελούνται στον server!

Τι είναι η PHP;

- Η PHP είναι ένα ακρωνύμιο προερχόμενο από τις λέξεις: "**P**HP **H**ypertext **P**reprocessor".
- Η PHP είναι μια ευρέως χρησιμοποιούμενη, ανοιχτού κώδικα scripting γλώσσα (γλώσσα για ανάπτυξη «σεναρίων»).
- Τα PHP scripts εκτελούνται στον server.
- Η PHP δεν κοστίζει τίποτα, μπορείτε να βρείτε εκτελέσιμα αρχεία και εφαρμογές και να τα χρησιμοποιήσετε δωρεάν.

Η PHP είναι μια καταπληκτική και δημοφιλής γλώσσα!

Είναι αρκετά ισχυρή για να είναι στο επίκεντρο του μεγαλύτερου συστήματος blogging στο διαδίκτυο (WordPress)!
Είναι τόσο σημαντική ώστε να «τρέχει» σε php το μεγαλύτερο κοινωνικό δίκτυο (facebook)!
Είναι επίσης αρκετά εύκολο για έναν αρχάριο να τη μάθει ως πρώτη server scripting γλώσσα ανάμεσα σε άλλες!

Τι είναι ένα PHP αρχείο;

- Τα PHP αρχεία μπορεί να περιέχουν κείμενο (text), HTML, CSS, JavaScript και PHP κώδικα.
- Ο PHP κώδικας εκτελείται στον server και το αποτέλεσμα επιστρέφεται στον browser ως απλό HTML.
- Τα PHP αρχεία έχουν επέκταση ".php"

Τι μπορεί να κάνει η PHP;

- Η PHP μπορεί να παράγει δυναμικό περιεχόμενο της σελίδας
- Η PHP μπορεί να δημιουργεί, να ανοίγει, να διαβάζει, να γράφει, να διαγράφει, και να κλείνει αρχεία στο διακομιστή (server).
- Η PHP μπορεί να συλλέγει δεδομένα φόρμας
- Η PHP μπορεί να στέλνει και να λαμβάνει cookies
- Η PHP μπορεί να προσθέσει, να διαγράψει, να τροποποιήσει τα δεδομένα στη βάση δεδομένων σας
- Η PHP μπορεί να περιορίσει ή να εμποδίσει την πρόσβαση χρηστών σε ορισμένες σελίδες του ιστοτόπού σας (website), εφόσον το επιθυμείτε.
- Η PHP μπορεί να κρυπτογραφήσει τα δεδομένα.

Με την PHP δεν είστε περιορισμένοι απλώς και μόνο να παράξετε HTML. Μπορείτε να εξάγετε εικόνες, αρχεία PDF, ακόμη και ταινίες Flash. Μπορείτε επίσης να παράξετε οποιοδήποτε κείμενο όπως XHTML και XML.

Γιατί PHP;

- Η PHP «τρέχει» σε διάφορες πλατφόρμες (Windows, Linux, Unix, Mac OS X, κλπ).
- Η PHP είναι συμβατή με σχεδόν όλους τους διακομιστές (servers) που χρησιμοποιούνται σήμερα (Apache, IIS, κλπ).
- Η PHP υποστηρίζει ένα ευρύ φάσμα των βάσεων δεδομένων.
- Η PHP είναι δωρεάν. Κατεβάστε την πλατφόρμα της PHP από την επίσημη πηγή PHP: www.php.net
- Η PHP είναι εύκολο να τη μάθετε και «τρέχει» στην πλευρά του διακομιστή.

PHP - Εγκατάσταση

Τι χρειάζομαι;

Για να αρχίσετε να χρησιμοποιείτε την PHP, μπορείτε να:

- Βρείτε έναν web-host («οικοδεσπότη ιστοσελίδων») με PHP και MySQL υποστήριξη (όπως ο EasyPHP).
- Εγκαταστήσετε έναν web server στο δικό σας υπολογιστή και στη συνέχεια να εγκαταστήσετε την PHP και MySQL.

Χρησιμοποιήστε έναν web host με PHP υποστήριξη

Αν ο διακομιστής σας έχει ενεργοποιηθεί για υποστήριξη της PHP δεν χρειάζεται να κάνετε τίποτα.

Απλά δημιουργήστε κάποια .php αρχεία, τοποθετήστε τα στον κατάλογο Ιστού σας (web directory), και ο server θα τα κάνει αυτόματα «parse» (ανάλυση και μετατροπή τους σε κατάλληλη μορφή) για εσάς.

Δεν χρειάζεται να μεταγλωττίσετε τίποτα ή να εγκαταστήσετε κάποια επιπλέον εργαλεία.

Επειδή η PHP είναι δωρεάν, οι περισσότεροι web hosts προσφέρουν υποστήριξη PHP.

Εγκατάσταση και ρύθμιση PHP για το δικό σας PC

Αν ο διακομιστής σας δεν υποστηρίζει PHP, θα πρέπει:

- Να εγκαταστήσετε ένα web server
- Να εγκαταστήσετε την PHP
- Να εγκαταστήσετε μια βάση δεδομένων, όπως η MySQL

Η επίσημη ιστοσελίδα της PHP (PHP.net) έχει οδηγίες εγκατάστασης για την PHP: http://php.net/manual/en/install.php

PHP - Σύνταξη

Το σενάριο PHP εκτελείται στον server, και το απλό αποτέλεσμα HTML στέλνεται πίσω στον browser.

Βασική σύνταξη PHP

Ένα σενάριο PHP μπορεί να τοποθετηθεί οπουδήποτε στο έγγραφο. Ένα σενάριο PHP ξεκινά με **<?php** και τελειώνει με **?>** :

```
<?php
// PHP code goes here
?>
```

Η προεπιλεγμένη επέκταση αρχείου για τα αρχεία PHP είναι ".php". Ένα αρχείο PHP κανονικά περιέχει ετικέτες HTML, και λίγο κώδικα PHP scripting.

Παρακάτω, έχουμε και άλλο παράδειγμα απλού PHP αρχείου, με ένα PHP script που χρησιμοποιεί μια ενσωματωμένη λειτουργία PHP "ηχούς" για την έξοδο του κειμένου "Καλημέρα World!" σε μια ιστοσελίδα. Σημειώστε ότι για την εμφάνιση Ελληνικών χαρακτήρων, εναλλακτικά της χρήσης <meta> μπορούμε να πάμε στο αρχείο php.ini και να δηλώσουμε default_charset = "UTF-8"

Παράδειγμα php_2.php

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
 </head>
```

```
<body>

  <h1>My first PHP σελίδα</h1>

 <?php
  echo "Γειά σου World!";
  ?>

</body>
</html>
```

Έξοδος:

My first PHP σελίδα

Γειά σου World!

Σημείωση: Οι δηλώσεις PHP τερματίζονται με το ελληνικό ερωτηματικό (;). Η ετικέτα κλεισίματος ενός μπλοκ του κώδικα PHP, επίσης, συνεπάγεται αυτόματα ένα ερωτηματικό (?).

Σχόλια σε PHP

Ένα σχόλιο σε κώδικα PHP είναι μια γραμμή που δεν διαβάζεται / εκτελείται ως μέρος του προγράμματος. Μοναδικός σκοπός του είναι να διαβαστεί από κάποιον που επεξεργάζεται τον κώδικα, αποτελεί λοιπόν τεκμηρίωση του κώδικα!

Τα σχόλια είναι χρήσιμα για:

- Για να επιτρέψετε σε άλλους να καταλάβουν τι κάνετε – Τα σχόλια επιτρέπουν σε άλλους προγραμματιστές να καταλάβουν τι είχατε κάνει σε κάθε βήμα (αν εργάζεστε σε μια ομάδα)
- Για να σας υπενθυμίζουν τι ακριβώς κάνατε - Οι περισσότεροι έμπειροι προγραμματιστές, επανερχόμενοι στη δουλειά τους ένα χρόνο ή δύο χρόνια αργότερα χρειάζεται να ξαναθυμηθούν και να καταλάβουν τι έκαναν τότε. Τα σχόλια μπορούν να σας υπενθυμίσουν αυτό που σκεφτόσασταν τη στιγμή που γράφατε τον κώδικα.

Η PHP υποστηρίζει τρεις τρόπους σχολιασμού:

Παράδειγμα

```
<!DOCTYPE html>
<html>
<body>
```

```
<?php
// This is a single line comment

# This is also a single line comment

/*
This is a multiple lines comment block
that spans over more than
one line
*/

echo "Hello World!";
?>

</body>
</html>
```

Έξοδος: Hello World!

PHP - Case Sensitivity

Στην PHP, όλες οι καθορισμένες από το χρήστη λειτουργίες, κλάσεις, λέξεις-κλειδιά (πχ if, else, while, echo, κλπ) ΔΕΝ ΕΙΝΑΙ case sensitive, δηλαδή είτε κάποια λέξη γράφεται με πεζά είτε με κεφαλαία γράμματα είναι ακριβώς το ίδιο και δεν επηρεάζει την εκτέλεση ενός php script.

Στο παράδειγμα που ακολουθεί και οι τρεις δηλώσεις echo είναι επιτρεπτές και παράγουν ακριβώς το ίδιο αποτέλεσμα:

Παράδειγμα

```
<!DOCTYPE html>
<html>
<body>

<?php
ECHO "Hello World!<br>";
echo "Hello World!<br>";
EcHo "Hello World!<br>";
?>

</body>
</html>
```

Έξοδος:
Hello World!
Hello World!
Hello World!

Όμως, στην PHP ΟΛΕΣ οι μεταβλητές είναι case sensitive.

Στο παρακάτω παράδειγμα, μόνο η πρώτη δήλωση θα εμφανίσει την τιμή της μεταβλητής $color (αυτό συμβαίνει επειδή οι μεταβλητές $color, $COLOR, $coLOR εκτελούνται ως τρεις διαφορετικές μεταβλητές):

Παράδειγμα php_3.php

```
<!DOCTYPE html>
<html>
<body>

<?php
$color="red";
echo "My car is " . $color . "<br>";
echo "My house is " . $COLOR . "<br>";
echo "My boat is " . $coLOR . "<br>";
?>

</body>
</html>
```

Έξοδος:
My car is red

Notice: Undefined variable: COLOR in **C:\Program Files\EasyPHP-DevServer-14.1VC11\data\localweb\Askiseis\PHP_3.php** on line **8**
My house is

Notice: Undefined variable: coLOR in **C:\Program Files\EasyPHP-DevServer-14.1VC11\data\localweb\Askiseis\PHP_3.php** on line **9**
My boat is

PHP - μεταβλητές

Οι μεταβλητές είναι containers («δοχεία») για την αποθήκευση πληροφοριών:

Παράδειγμα php_4.php

```
<!DOCTYPE html>
<html>
<body>

<?php
```

```
$x=5;
$y=6;
$z=$x+$y;
echo $z;
?>

</body>
</html>
```

Έξοδος:
11

Μοιάζει με Άλγεβρα

x = 5
y = 6
z = x + y

Στην άλγεβρα χρησιμοποιούμε γράμματα (όπως x) για να αποθηκεύουν ή να «κρατούν» τιμές (όπως το 5).

Από την έκφραση z=x + y παραπάνω, μπορούμε να υπολογίσουμε την τιμή του z να είναι 11.

Στην PHP αυτά τα γράμματα ονομάζονται **μεταβλητές.**

Μεταβλητές της PHP

Όπως και με την άλγεβρα, οι μεταβλητές της PHP χρησιμοποιούνται για να «κρατούν» απλές τιμές (x = 5) ή πιο σύνθετες εκφράσεις (z = x + y).

Μια μεταβλητή μπορεί να έχει ένα σύντομο όνομα (όπως το X και Y) ή ένα πιο περιγραφικό όνομα (age, carname, total_volume).

Κανόνες για τις μεταβλητές της PHP:

- Μια μεταβλητή ξεκινά με το σύμβολο $, που ακολουθείται από το όνομα της μεταβλητής
- Το όνομα μιας μεταβλητής πρέπει να αρχίζει με ένα γράμμα ή το χαρακτήρα της «κάτω παύλας»

- Το όνομα μιας μεταβλητής δεν μπορεί να ξεκινήσει με έναν αριθμό
- Ένα όνομα μεταβλητής μπορεί να περιέχει μόνο αλφαριθμητικούς χαρακτήρες και κάτω παύλες (AZ, 0-9 και _)
- Τα ονόματα των μεταβλητών είναι case sensitive ($y και $Y είναι δύο διαφορετικές μεταβλητές)

 Να θυμάστε ότι τα ονόματα των μεταβλητών της PHP είναι case-sensitive!

Δημιουργία (Δηλώσεις) μεταβλητών της PHP

Η PHP δεν έχει καμία εντολή για τη δήλωση μιας μεταβλητής.

Μια μεταβλητή δημιουργείται την πρώτη στιγμή που θα εκχωρήσετε μια τιμή σε αυτήν:

Παράδειγμα

```
<?php
$txt="Hello world!";
$x=5;
$y=10.5;
?>
```

Μετά την εκτέλεση των παραπάνω καταστάσεων, η μεταβλητή **txt** θα κρατήσει την τιμή **Hello world!** , η μεταβλητή **x** θα κρατήσει την τιμή **5** και η μεταβλητή **y** θα κρατήσει την τιμή **10.5**.

Σημείωση: Όταν εκχωρείτε μια τιμή κειμένου σε μια μεταβλητή, βάλτε εισαγωγικά γύρω από την τιμή.

Η PHP είναι μια Γλώσσα «χαλαρής δήλωσης τύπου» **(Loosely Typed)**

Στο παραπάνω παράδειγμα, παρατηρούμε ότι εμείς δεν χρειάζεται να πούμε στην PHP κάτι για το ποιος είναι ο τύπος δεδομένων της μεταβλητής.

Η PHP αυτόματα μετατρέπει την μεταβλητή στον σωστό τύπο δεδομένων, ανάλογα με την αξία του.

Σε άλλες γλώσσες, όπως η C, C ++, και Java, ο προγραμματιστής πρέπει να δηλώσει το όνομα και τον τύπο της μεταβλητής πριν το χρησιμοποιήσει.

Εμβέλεια και πεδία (Scope) των μεταβλητών της PHP

Στην PHP, μεταβλητές μπορούν να δηλωθούν οπουδήποτε στο σενάριο (script).

Το πεδίο μιας μεταβλητής είναι το μέρος του σεναρίου, όπου η μεταβλητή μπορεί να χρησιμοποιείται και να έχει ισχύ ή να «δραστηριοποιείται».

Η PHP έχει τρία διαφορετικά πεδία μεταβλητών (πεδία «δραστηριότητας» των μεταβλητών):

- καθολικές μεταβλητές
- τοπικές μεταβλητές
- στατικές μεταβλητές

Τοπική και καθολική εμβέλεια

Μια μεταβλητή που δηλώνεται **έξω από** μια συνάρτηση έχει καθολική εμβέλεια και μπορεί να προσεγγιστεί μόνο εκτός συνάρτησης.

Μια μεταβλητή που δηλώνεται **μέσα σε** μια συνάρτηση έχει τοπική εμβέλεια και μπορεί να προσεγγιστεί μόνο μέσα σε αυτή τη συνάρτηση.

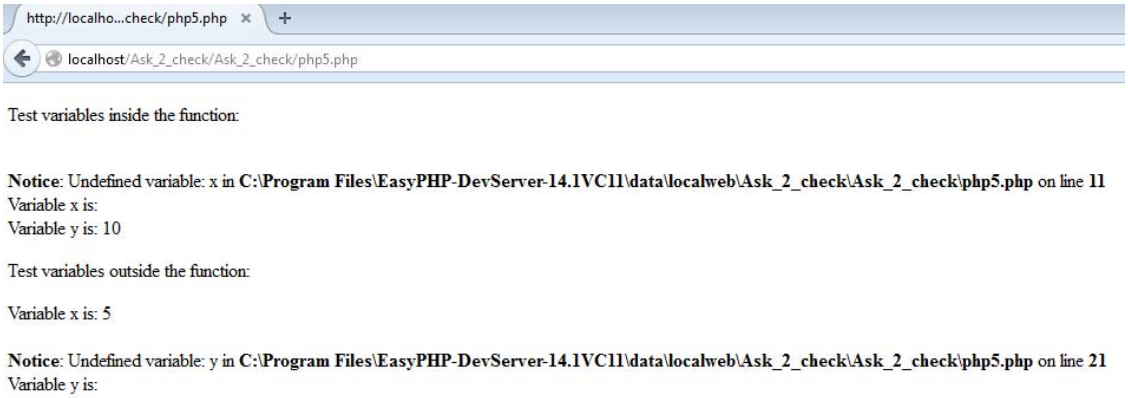
Το παρακάτω παράδειγμα ελέγχει μεταβλητές με τοπική και καθολική εμβέλεια:

Παράδειγμα php_5.php

```
<!DOCTYPE html>
<html>
<body>

<?php
$x=5; // global scope

function myTest() {
  $y=10; // local scope
  echo "<p>Test variables inside the function:</p>";
  echo "Variable x is: $x";
  echo "<br>";
  echo "Variable y is: $y";
}

myTest();
```

```
echo "<p>Test variables outside the function:</p>";
echo "Variable x is: $x";
echo "<br>";
echo "Variable y is: $y";
?>

</body>
</html>
```

Έξοδος:



Στο παραπάνω παράδειγμα υπάρχουν δύο μεταβλητές $x και $y και μια συνάρτηση myTest(). Η $x είναι μια καθολική μεταβλητή αφού έχει δηλωθεί εκτός συνάρτησης και η $y είναι μια τοπική μεταβλητή, δεδομένου ότι δημιουργείται στο εσωτερικό της συνάρτησης.

Όταν δούμε στην έξοδο τις τιμές των δύο μεταβλητών που βρίσκονται μέσα στη συνάρτηση myTest(), παρατηρούμε πως τυπώνει την τιμή της $y, αφού αυτή έχει δηλωθεί τοπικά, αλλά δεν μπορεί να τυπώσει την τιμή της $x, δεδομένου ότι δημιουργείται έξω από τη συνάρτηση myTest() .

Στη συνέχεια, όταν πάρουμε στην έξοδο τις τιμές των δύο μεταβλητών εκτός συνάρτησης myTest(), τυπώνεται η τιμή της $x, αλλά δεν μπορεί να τυπώσει την τιμή της $y, δεδομένου ότι είναι μια τοπική μεταβλητή και έχει δημιουργηθεί στο εσωτερικό της συνάρτησης myTest().

Μπορείτε να έχετε τοπικές μεταβλητές με το ίδιο όνομα σε διαφορετικές συναρτήσεις, επειδή οι τοπικές μεταβλητές αναγνωρίζονται μόνο από τη συνάρτηση στην οποία έχουν δηλωθεί.

PHP - η λέξη-κλειδί «global»

Η λέξη-κλειδί ***global*** χρησιμοποιείται για να αποκτήσετε πρόσβαση σε μια καθολική μεταβλητή μέσα από μια συνάρτηση.

Για να το κάνετε αυτό, χρησιμοποιήστε τη λέξη-κλειδί ***global*** πριν από τις μεταβλητές (μέσα στη συνάρτηση):

Παράδειγμα php_6.php

```
<?php
$x=5;
$y=10;

function myTest() {
  global $x, $y;
  $y=$x+$y;
}

myTest();
echo $y; // outputs 15
?>
```

Η PHP αποθηκεύει επίσης όλες τις καθολικές μεταβλητές σε έναν πίνακα (array) που ονομάζεται $GLOBALS[*δείκτης*]. Ο *δείκτης (index)* «κρατάει» το όνομα της μεταβλητής. Αυτός ο πίνακας είναι επίσης προσβάσιμος μέσα από τις συναρτήσεις και μπορεί να χρησιμοποιηθεί για να ενημερώσετε άμεσα καθολικές μεταβλητές.

Το παραπάνω παράδειγμα μπορεί να ξαναγραφτεί ως εξής:

Παράδειγμα php_7.php

```
<?php
$x=5;
$y=10;

function myTest() {
  $GLOBALS['y']=$GLOBALS['x']+$GLOBALS['y'];
}

myTest();
echo $y; // outputs 15
?>
```

PHP - η λέξη κλειδί «static»

Κανονικά, όταν μια συνάρτηση έχει ολοκληρωθεί/εκτελεσθεί, το σύνολο των μεταβλητών της διαγράφονται. Ωστόσο, μερικές φορές θέλουμε μια τοπική μεταβλητή να ΜΗΝ διαγραφεί. Τη χρειαζόμαστε για περαιτέρω εργασία.

Για να το κάνετε αυτό, χρησιμοποιήστε τη λέξη-κλειδί ***static*** όταν αρχικά δηλώνετε τη μεταβλητή:

Παράδειγμα php_8.php

```
<?php

function myTest() {
  static $x=0;
  echo $x;
  $x++;
}

myTest();
echo "<br>";
myTest();
echo "<br>";
myTest();
echo "<br>";

?>
```

Το αποτέλεσμα είναι:

```
0
1
2
```

Δηλαδή, κάθε φορά που καλείται η συνάρτηση, η συγκεκριμένη μεταβλητή ($x) θα εξακολουθεί να έχει τις πληροφορίες που περιείχε την τελευταία φορά (όταν είχε κληθεί η συνάρτηση).

Σημείωση: Η μεταβλητή static εξακολουθεί να είναι τοπική στη συνάρτηση.

PHP - echo και print δηλώσεις

Στην PHP υπάρχουν δύο βασικοί τρόποι για να πάρετε εξόδους: echo και print.

Χρησιμοποιούμε echo (και εκτύπωση), σχεδόν σε κάθε παράδειγμα. Έτσι, το κεφάλαιο αυτό περιέχει λίγο περισσότερες πληροφορίες σχετικά με αυτές τις δύο καταστάσεις εξόδου.

PHP δηλώσεις echo και print

Υπάρχουν ορισμένες διαφορές μεταξύ της echo και της print:

- echo - μπορεί να εξάγει ένα ή περισσότερα αλφαριθμητικά (strings)
- print - μπορεί να εξάγει μόνο ένα αλφαριθμητικό, και επιστρέφει πάντα 1

Σημείωση: Η echo είναι οριακά πιο γρήγορη σε σύγκριση με την print, καθώς η echo δεν επιστρέφει καμία αξία.

Η δήλωση echo της PHP

Η echo είναι μια γλωσσική κατασκευή και μπορεί να χρησιμοποιηθεί με ή χωρίς παρένθεση: echo ή echo().

Εμφάνιση αλφαριθμητικών

Το παρακάτω παράδειγμα δείχνει πώς μπορείτε να εμφανίσετε διαφορετικά αλφαριθμητικά με την εντολή echo (επίσης να παρατηρήσετε ότι τα αλφαριθμητικά μπορεί να περιέχουν σήμανση HTML):

Παράδειγμα php_9.php

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
</head>
```

```
<body>

<?php
echo "<h2>PHP is fun!</h2>";
echo "Καλημέρα!<br>";
echo "I'm about to learn PHP!<br>";
echo "This", " string", " was", " made", " with multiple parameters.";
?>

</body>
</html>
```

Έξοδος:

PHP is fun!

Καλημέρα!
I'm about to learn PHP!
This string was made with multiple parameters.

Εμφάνιση μεταβλητών

Το παρακάτω παράδειγμα δείχνει πώς μπορείτε να εμφανίσετε αλφαριθμητικά και μεταβλητές με την εντολή echo:

Παράδειγμα php_10.php

```
<!DOCTYPE html>
<html>
<body>

<?php
$txt1="Learn PHP";
$txt2="W3Schools.com";
$cars=array("Ford","BMW","Toyota");
echo $txt1;
echo "<br>";
echo "Study PHP at $txt2";
echo "<br>";
echo "My car is a {$cars[0]}";
?>

</body>
</html>
```

Έξοδος:

Learn PHP
Study PHP at W3Schools.com
My car is a Ford

Η δήλωση print της PHP

Η print είναι επίσης γλωσσική κατασκευή, και μπορεί να χρησιμοποιηθεί με ή χωρίς παρένθεση: print ή print().

Εμφάνιση αλφαριθμητικών

Το παρακάτω παράδειγμα δείχνει πώς μπορείτε να εμφανίσετε διαφορετικά αλφαριθμητικά με την εντολή print (επίσης να παρατηρήσετε ότι τα αλφαριθμητικά μπορεί να περιέχουν σήμανση HTML):

Παράδειγμα (php_11.php)

```
<!DOCTYPE html>
<html>
<body>

<?php
print "<h2>PHP is fun!</h2>";
print "Hello world!<br>";
print "I'm about to learn PHP!";
?>

</body>
</html>
```

Έξοδος:

PHP is fun!

Hello world!
I'm about to learn PHP!

Εμφάνιση μεταβλητών

Το παρακάτω παράδειγμα δείχνει πώς μπορείτε να εμφανίσετε αλφαριθμητικά και μεταβλητές με την εντολή print:

Παράδειγμα (php_12.php)

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

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

<body>
<?php
$txt1="Learn PHP";
$txt2="ΠαΜακ";
$cars=array("Volvo","BMW","Toyota");
print $txt1;
print "<br>";
print "Study PHP at $txt2";
print "<br>";
print "My car είναι {$cars[0]}";
?>
</body>
</html>
```

Έξοδος:
Learn PHP
Study PHP at ΠαΜακ
My car είναι Volvo

PHP - Τύποι Δεδομένων

- String
- Integer
- Float (floating point numbers - also called double)
- Boolean
- Array
- Object
- NULL
- Resource

Δηλαδή: αλφαριθμητικό ή συμβολοσειρά, ακέραιος, αριθμός κινητής υποδιαστολής, λογικός, πίνακας, αντικείμενο, NULL, πόρος.

PHP String

Ένα string ή μια συμβολοσειρά είναι μια ακολουθία από χαρακτήρες, όπως "Γειά σου Κόσμε!".

Μια συμβολοσειρά μπορεί να είναι οποιοδήποτε κείμενο μέσα σε εισαγωγικά. Μπορείτε να χρησιμοποιήσετε μονά ή διπλά εισαγωγικά:

Παράδειγμα

```
<?php
$x = "Hello world!";
echo $x;
echo "<br>";
$x = 'Hello world!';
echo $x;
?>
```

PHP Integer

Ένας ακέραιος είναι ένας αριθμός χωρίς δεκαδικά ψηφία.

Κανόνες για ακέραιους αριθμούς:

- Ένας ακέραιος πρέπει να έχει τουλάχιστον ένα ψηφίο (0-9)
- Ένας ακέραιος δεν μπορεί να περιέχει κόμμα ή κενά
- Ένας ακέραιος δεν πρέπει να έχει δεκαδικό ψηφίο
- Ένας ακέραιος αριθμός μπορεί να είναι είτε θετικός είτε αρνητικός
- Ακέραιοι μπορούν να οριστούν σε τρεις μορφές: δεκαδικοί (10-based), δεκαεξαδικοί (16-based - με πρόθεμα 0x) ή οκταδικοί (8-βάση - με πρόθεμα 0)

Στο ακόλουθο παράδειγμα, θα δοκιμάσουμε διαφορετικούς αριθμούς. Η συνάρτηση var_dump() επιστρέφει τον τύπο και την τιμή των μεταβλητών:

Παράδειγμα (php_13.php)

```
<!DOCTYPE html>
<html>
<body>

<?php
$x = 5985;
var_dump($x);
echo "<br>";
$x = -345; // negative number
var_dump($x);
echo "<br>";
$x = 0x8C; // hexadecimal number
var_dump($x);
echo "<br>";
```

```
$x = 047; // octal number
var_dump($x);
?>

</body>
</html>
```

Έξοδος:
int(5985)
int(-345)
int(140)
int(39)

PHP Float - Αριθμός κινητής υποδιαστολής (Floating Point)

Ένας αριθμός κινητής υποδιαστολής είναι ένας αριθμός με μια δεκαδική υποδιαστολή ή ένας αριθμό σε εκθετική μορφή.

Στο ακόλουθο παράδειγμα, θα δοκιμάσουμε διαφορετικούς αριθμούς. Η συνάρτηση var_dump() επιστρέφει τον τύπο και την τιμή των μεταβλητών:

Παράδειγμα (php_14.php)

```
<!DOCTYPE html>
<html>
<body>

<?php
$x = 10.365;
var_dump($x);
echo "<br>";
$x = 2.4e3;
var_dump($x);
echo "<br>";
$x = 8E-5;
var_dump($x);
?>

</body>
</html>
```

Έξοδος:
float(10.365)
float(2400)
float(8.0E-5)

PHP Boolean

Ο τύπος Boolean αναπαριστά δυο πιθανές καταστάσεις: true (αλήθεια) ή false (ψέμμα).

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

Χρησιμοποιούνται συχνά σε έλεγχο υποθέσεων. Θα μάθετε περισσότερα για τον έλεγχο υποθέσεων σε επόμενο κεφάλαιο.

PHP Array (πίνακας)

Ένας πίνακας αποθηκεύει πολλαπλές τιμές σε μια μόνο μεταβλητή.

Στο παρακάτω παράδειγμα δημιουργούμε έναν πίνακα, και στη συνέχεια χρησιμοποιούμε τη συνάρτηση var_dump() για να επιστρέψει τον τύπο και την τιμή του πίνακα:

Παράδειγμα

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

Έξοδος:

```
array(3) { [0]=> string(5) "Volvo" [1]=> string(3) "BMW" [2]=> string(6) "Toyota" }
```

Θα μάθετε πολλά περισσότερα για πίνακες στα επόμενα κεφάλαια.

PHP Object (αντικείμενο)

Ένα αντικείμενο είναι ένας τύπος δεδομένων που αποθηκεύει δεδομένα και πληροφορίες σχετικά με τον τρόπο επεξεργασίας των εν λόγω δεδομένων.

Στην PHP, ένα αντικείμενο, πρέπει να δηλώνεται ρητά.

Πρώτα πρέπει να δηλώσουμε μια κλάση του αντικειμένου. Για το σκοπό αυτό, χρησιμοποιούμε τη λέξη-κλειδί ***class***. Μια κλάση είναι μια δομή που μπορεί να περιέχει

ιδιότητες και μεθόδους. Μπορούμε στη συνέχεια να ορίσουμε τον τύπο δεδομένων στην κλάση του αντικειμένου, και στη συνέχεια χρησιμοποιούμε τον τύπο δεδομένων σε στιγμιότυπα αυτής της κλάσης:

Παράδειγμα

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

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

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

Έξοδος:
VW

Θα μάθετε περισσότερα για τα αντικείμενα σε επόμενο κεφάλαιο.

PHP - τιμή NULL

Η null είναι ένας ειδικός τύπος δεδομένων που μπορεί να έχει μόνο μια τιμή: NULL.

Μια μεταβλητή τύπου NULL είναι μια μεταβλητή στην οποία δεν έχει καμία τιμή εκχωρηθεί. Αν μια μεταβλητή δημιουργηθεί χωρίς να της δοθεί τιμή, αυτόματα παίρνει τη τιμή NULL.

Οι μεταβλητές μπορούν να ‘εκκενωθούν’ (να μην έχουν τιμή) εάν θέσετε στη μεταβλητή την τιμή NULL:

Παράδειγμα

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

Έξοδος:
NULL

PHP Resource

Ο ειδικός αυτός τύπος δεν είναι ακριβώς τύπος δεδομένων. Είναι η αποθήκευση μιας αναφοράς σε συναρτήσεις και πόρους εξωτερικούς της PHP. Ένα κοινό παράδειγμα χρήσης του resource τύπου είναι μια κλήση προς βάση δεδομένων.

PHP - Συναρτήσεις String

Μια συμβολοσειρά (string) είναι μια ακολουθία από χαρακτήρες, όπως "Γειά σου Κόσμε!".

PHP συναρτήσεις συμβολοσειράς/string

Σε αυτό το κεφάλαιο θα δούμε κάποιες κοινώς χρησιμοποιούμενες συναρτήσεις χειρισμού strings.

Η PHP συνάρτηση strlen ()

Η συνάρτηση strlen() επιστρέφει το μήκος μιας συμβολοσειράς, σε χαρακτήρες.

Το παρακάτω παράδειγμα επιστρέφει το μήκος του string "Γειά σου Κόσμε!":

Παράδειγμα (php_15.php)

```
<?php
echo strlen("Hello world!");
?>
```

Η έξοδος του κώδικα παραπάνω θα είναι: 12

Συμβουλή: Η strlen() χρησιμοποιείται συχνά σε βρόχους ή άλλες συναρτήσεις, όταν είναι σημαντικό να γνωρίζουμε πότε τελειώνει ένα string. Δηλαδή σε ένα βρόχο, ίσως να θέλουμε να σταματήσουμε το βρόχο μετά τον τελευταίο χαρακτήρα σε ένα string.

Καταμέτρηση του πλήθους των λέξεων σε ένα String

Η PHP συνάρτηση str_word_count()μετρά το πλήθος των λέξεων σε ένα string:

Παράδειγμα

```
<?php
echo str_word_count("Hello world!"); // outputs 2
?>
```

Αντιστροφή String

Η PHP συνάρτηση strrev() αντιστρέφει ένα string:

Παράδειγμα

```
<?php
echo strrev("Hello world!"); // outputs !dlrow olleH
?>
```

Η συνάρτηση PHP strpos ()

Η συνάρτηση strpos() χρησιμοποιείται για να ψάξει για ένα συγκεκριμένο χαρακτήρα ή κείμενο μέσα σε ένα string. Εάν βρεθεί μια αντιστοιχία, θα επιστρέψει τη θέση του χαρακτήρα της πρώτης αντιστοίχισης.

Το παράδειγμα παρακάτω αναζητά τη λέξη "world" στη συμβολοσειρά "Hello world!":

Παράδειγμα (php_16.php)

```
<?php
echo strpos("Hello world!","world");
?>
```

Η έξοδος του κώδικα πάνω θα είναι: 6.

Η θέση της συμβολοσειράς "world" στο παραπάνω παράδειγμα είναι 6. Ο λόγος που είναι 6 (και όχι 7), είναι ότι η πρώτη θέση χαρακτήρα στο string είναι 0, και όχι 1.

Αντικατάσταση κειμένου μέσα σε String

Η PHP συνάρτηση str_replace() αντικαθιστά χαρακτήρες ενός string με άλλους χαρακτήρες.

Παράδειγμα

```
<?php
echo str_replace("world", "Dolly", "Hello world!"); // outputs Hello Dolly!
?>
```

Πλήρης αναφορά PHP συναρτήσεων String

Για μια πλήρη αναφορά όλων των συναρτήσεων string, πηγαίνετε στην σελίδα http://www.w3schools.com/php/php_ref_string.asp

PHP - Σταθερές

Οι σταθερές είναι σαν τις μεταβλητές εκτός από το ότι από τη στιγμή που ορίζονται, δεν μπορούν να αλλάξουν ή να επαναπροσδιοριστούν.

Μια σταθερά είναι ένα αναγνωριστικό (όνομα) για μια απλή τιμή. Η τιμή δεν μπορεί να μεταβληθεί κατά τη διάρκεια του σεναρίου. Ένα έγκυρο όνομα σταθεράς αρχίζει με ένα γράμμα ή underscore (κανένα σύμβολο $ πριν από το όνομα σταθεράς). **Σημείωση:** Σε αντίθεση με τις μεταβλητές, οι σταθερές έχουν αυτόματα καθολική ισχύ σε ολόκληρο το σενάριο.

Ορίστε μια PHP Σταθερά

Για να ορίσετε μια σταθερά, χρησιμοποιήστε τη συνάρτηση define () - παίρνει τρεις παραμέτρους: Η πρώτη παράμετρος καθορίζει το όνομα της σταθεράς, η δεύτερη παράμετρος καθορίζει την τιμή της σταθεράς, και η προαιρετική τρίτη παράμετρος καθορίζει το αν το όνομα της σταθεράς θα πρέπει να είναι case -insensitive. Προεπιλογή είναι ψευδής.

define(*name*, *value*, *case-insensitive*)

Το παρακάτω παράδειγμα δημιουργεί μια **case-sensitive σταθερά**, με την τιμή του "Καλώς ήρθατε στο W3Schools.com!":

Παράδειγμα (php_17.php)

```
<!DOCTYPE html>
<html>
<body>

<?php
// define a case-sensitive constant
define("GREETING", "Welcome to W3Schools.com!");
echo GREETING;
echo "<br>";
// will not output the value of the constant
echo greeting;
?>

</body>
</html>
```

Έξοδος:

Welcome to W3Schools.com!

Notice: Use of undefined constant greeting - assumed 'greeting' in **C:\Program Files\EasyPHP-DevServer-14.1VC11\data\localweb\Ask_2_check\Ask_2_check\php_17.php** on line **11**
greeting

Το παρακάτω παράδειγμα δημιουργεί μια **case-insensitive σταθερά**, με την τιμή του "Καλώς ήρθατε στο W3Schools.com!": (*) Στις νέες εκδόσεις PHP, δεν υποστηρίζεται πλέον. Οι σταθερές είναι μόνο case-sensitive

Παράδειγμα (php_18.php)

```
<!DOCTYPE html>
<html>
<body>

<?php
// define a case-insensitive constant
define("GREETING", "Welcome to W3Schools.com!", true);
echo GREETING;
echo "<br>";
// will also output the value of the constant
echo greeting;
?>

</body>
</html>
```

Οι Σταθερές είναι ολικής εμβέλειας (Global)

Οι σταθερές είναι αυτόματα global και μπορούν να χρησιμοποιηθούν παντού.

Παράδειγμα

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

function myTest() {
    echo GREETING;
}

myTest();
?>
```

PHP - Τελεστές

Οι τελεστές χρησιμοποιούνται για να γίνουν πράξεις σε μεταβλητές και τιμές. Η PHP διακρίνει τις ακόλουθες κατηγορίες:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Increment/Decrement operators
- Logical operators
- String operators
- Array operators

PHP Αριθμητικοί Τελεστές

Operator	Name	Example	Result
+	Πρόσθεση	$x + $y	Sum of $x and $y
-	Αφαίρεση	$x - $y	Difference of $x and $y
*	Πολλαπλασιασμός	$x * $y	Product of $x and $y

/	Διαίρεση	$x / $y	Quotient of $x and $y (*)
%	Υπόλοιπο	$x % $y	Remainder of $x divided by $y
**	Δύναμη	$x ** $y	Result of raising $x to the $y'th power (Introduced in PHP 5.6)

(*) intdiv() για πηλίκο ακέραιας διαίρεσης

Το παρακάτω παράδειγμα δείχνει τα διαφορετικά αποτελέσματα από τη χρήση των διαφόρων αριθμητικών τελεστών:

Παράδειγμα (php_19.php)

```
<?php
  $x=10;
  $y=6;
  echo ($x + $y); // outputs 16
  echo ($x - $y); // outputs 4
  echo ($x * $y); // outputs 60
  echo ($x / $y); // outputs 1.6666666666667
  echo ($x % $y); // outputs 4
?>
```

PHP Τελεστές Ανάθεσης

Οι PHP τελεστές ανάθεσης χρησιμοποιούνται για να γράψουν μια τιμή σε μια μεταβλητή. Ο βασικός τελεστής ανάθεσης είναι "=". Αυτό σημαίνει ότι το αριστερό μέρος παίρνει την τιμή της έκφρασης ανάθεσης από το δεξί μέρος.

Ανάθεση	Είναι ίδια με...	Περιγραφή
x = y		Το αριστερό μέρος παίρνει την τιμή της έκφρασης ανάθεσης από το δεξί μέρος
x += y	x = x + y	Addition
x -= y	x = x - y	Subtraction
x *= y	x = x * y	Multiplication
x /= y	x = x / y	Division
x %= y	x = x % y	Modulus

Το παρακάτω παράδειγμα δείχνει τα διαφορετικά αποτελέσματα από τη χρήση των διαφόρων τελεστών ανάθεσης:

Παράδειγμα (php_20.php)

```
<?php
$x=10;
echo $x; // outputs 10

$y=20;
$y += 100;
echo $y; // outputs 120

$z=50;
$z -= 25;
echo $z; // outputs 25

$i=5;
$i *= 6;
echo $i; // outputs 30

$j=10;
$j /= 5;
echo $j; // outputs 2

$k=15;
$k %= 4;
echo $k; // outputs 3
?>
```

PHP Τελεστές Σύγκρισης

Οι PHP τελεστές σύγκρισης χρησιμοποιούνται για τη σύγκριση δύο τιμών (αριθμών ή string):

Operator	Name	Example	Result
==	Equal	$x == $y	True if $x is equal to $y
===	Identical	$x === $y	True if $x is equal to $y, and they are of the same type
!=	Not equal	$x != $y	True if $x is not equal to $y
<>	Not equal	$x <> $y	True if $x is not equal to $y
!==	Not identical	$x !== $y	True if $x is not equal to $y, or they are not of the same type
>	Greater than	$x > $y	True if $x is greater than $y
<	Less than	$x < $y	True if $x is less than $y

>=	Greater than or equal to	$x >= $y	True if $x is greater than or equal to $y
<=	Less than or equal to	$x <= $y	True if $x is less than or equal to $y

Το παρακάτω παράδειγμα δείχνει τα διαφορετικά αποτελέσματα της χρήσης κάποιων από τους τελεστές σύγκρισης:

Παράδειγμα (php_21.php)

```
<?php
$x=100;
$y="100";

var_dump($x == $y);
echo "<br>";
var_dump($x === $y);
echo "<br>";
var_dump($x != $y);
echo "<br>";
var_dump($x !== $y);
echo "<br>";

$a=50;
$b=90;

var_dump($a > $b);
echo "<br>";
var_dump($a < $b);
?>
```

Έξοδος:
bool(true)
bool(false)
bool(false)
bool(true)
bool(false)
bool(true)

PHP τελεστές αύξησης / μείωσης

Operator	Name	Description
++$x	Pre-increment	Increments $x by one, then returns $x
$x++	Post-increment	Returns $x, then increments $x by one
--$x	Pre-decrement	Decrements $x by one, then returns $x
$x--	Post-decrement	Returns $x, then decrements $x by one

Το παρακάτω παράδειγμα δείχνει τα διαφορετικά αποτελέσματα από τη χρήση των διαφόρων τελεστών αύξησης / μείωσης:

Παράδειγμα (php_22.php)

```
<?php
$x=10;
echo ++$x; // outputs 11

$y=10;
echo $y++; // outputs 10

$z=5;
echo --$z; // outputs 4

$i=5;
echo $i--; // outputs 5
?>
```

PHP λογικοί τελεστές

Operator	Name	Example	Result
and	And	$x and $y	True if both $x and $y are true
or	Or	$x or $y	True if either $x or $y is true
xor	Xor	$x xor $y	True if either $x or $y is true, but not both
&&	And	$x && $y	True if both $x and $y are true
||	Or	$x || $y	True if either $x or $y is true
!	Not	!$x	True if $x is not true

Παράδειγμα (php_23.php)

```
<!DOCTYPE html>
<html>
<body>

<?php
$x = 100;
$y = 50;

if ($x == 100 or $y == 80) {
    echo "Hello world!";
}
?>

</body>
</html>
```

Έξοδος:
Hello world!

PHP Τελεστές String

Operator	Name	Example	Result
.	Concatenation	$txt1 = "Hello" $txt2 = $txt1 . " world!"	Now $txt2 contains "Hello world!"
.=	Concatenation assignment	$txt1 = "Hello" $txt1 .= " world!"	Now $txt1 contains "Hello world!"

Το παρακάτω παράδειγμα δείχνει τα αποτελέσματα της χρήσης των string τελεστών:

Παράδειγμα (php_24.php)

```
<?php
$a = "Hello";
$b = $a . " world!";
echo $b; // outputs Hello world!

$x="Hello";
$x .= " world!";
echo $x; // outputs Hello world!
?>
```

PHP τελεστές πίνακα

Οι PHP τελεστές πίνακα (array) χρησιμοποιούνται για σύγκριση πινάκων:

Operator	Name	Example	Result
+	Union	$x + $y	Union of $x and $y (*)
==	Equality	$x == $y	True if $x and $y have the same key/value pairs
===	Identity	$x === $y	True if $x and $y have the same key/value pairs in the same order and of the same types
!=	Inequality	$x != $y	True if $x is not equal to $y
<>	Inequality	$x <> $y	True if $x is not equal to $y
!==	Non-identity	$x !== $y	True if $x is not identical to $y

(*) The + operator returns the right-hand array appended to the left-hand array; for keys that exist in both arrays, the elements from the left-hand array will be used, and the matching elements from the right-hand array will be ignored.

Το παρακάτω παράδειγμα δείχνει τα διαφορετικά αποτελέσματα από τη χρήση των διαφόρων τελεστών πίνακα:

Παράδειγμα (php_25.php)

```
<?php
 $x = array("a" => "red", "b" => "green");
 $y = array("c" => "blue", "d" => "yellow");
 $z = $x + $y; // union of $x and $y
 var_dump($z);
 var_dump($x == $y);
 var_dump($x === $y);
 var_dump($x != $y);
 var_dump($x <> $y);
 var_dump($x !== $y);  ?>
```

Έξοδος:

```
array(4) { ["a"]=> string(3) "red" ["b"]=> string(5) "green" ["c"]=> string(4) "blue" ["d"]=> string(6) "yellow" }
bool(false)
bool(false)
bool(true)
bool(true)
bool(true)
```

PHP – Εντολή If ... else ... elseif

Οι εντολές υπό όρους χρησιμοποιούνται για να εκτελέσουν διάφορες ενέργειες που βασίζονται σε διαφορετικές συνθήκες/υποθέσεις.

PHP εντολές υπό όρους

Πολύ συχνά όταν γράφετε κώδικα, θέλετε να εκτελέσετε διάφορες ενέργειες για διαφορετικές συνθήκες. Μπορείτε να χρησιμοποιήσετε τις υπό όρους εντολές στον κώδικά σας για να το κάνετε αυτό.

Στην PHP έχουμε τις ακόλουθες δηλώσεις υπό όρους:

- **If** - εκτελεί κάποιο κώδικα μόνο αν μια συγκεκριμένη συνθήκη είναι αληθής
- **if ... else** - εκτελεί κάποιο κώδικα, αν μια συνθήκη είναι αληθής και έναν άλλο κώδικα εάν η συνθήκη είναι ψευδής
- **if... elseif .... else** - επιλέγει ένα από τα πολλά μπλοκ του κώδικα που θα εκτελεστεί
- **switch** - επιλέγει ένα από τα πολλά μπλοκ του κώδικα που θα εκτελεστεί

PHP - Η εντολή if

Η εντολή if χρησιμοποιείται για να εκτελέσει τον κώδικα, **μόνο αν μια συγκεκριμένη συνθήκη είναι αληθής** .

Σύνταξη

if (*συνθήκη*) {
  κώδικας προς εκτέλεση εάν η συνθήκη είναι αληθής;
}

Το παρακάτω παράδειγμα θα δώσει έξοδο: "It's early...." εάν η τρέχουσα ώρα (“H”) είναι μεγαλύτερη από 6:

Παράδειγμα (php_26.php)

```
<?php
$t=date("H");

if ($t>"6") {
```

```
  echo "It's early....”
}
?>
```

PHP - Η if ... else εντολή

Χρησιμοποιήστε την if .... else εντολή για να εκτελέσετε έναν κώδικα, **αν μια συνθήκη είναι αληθής και έναν άλλο κώδικα, εάν η συνθήκη είναι ψευδής** .

Σύνταξη

```
if (συνθήκη) {
  κώδικας προς εκτέλεση εάν η συνθήκη είναι αληθής;} else {
    κώδικας προς εκτέλεση εάν η συνθήκη είναι ψευδής;
}
```

Το παρακάτω παράδειγμα θα δώσει έξοδο: "Να έχετε μια καλή μέρα!" εάν η τρέχουσα ώρα είναι μικρότερη από “19”, και "Να έχετε μια καλή νύχτα!" αλλιώς:

Παράδειγμα (php_27.php)

```
<?php
$t=date("H");

if ($t<"19") {
  echo "Να έχετε μια καλή μέρα!";
} else {
  echo "Να έχετε μια καλή νύχτα!";
}
?>
```

PHP – Η if ... elseif .... else εντολή

Χρησιμοποιήστε την για να **επιλέξετε ένα από τα πολλά μπλοκ του κώδικα που πρόκειται να εκτελεστεί** .

Σύνταξη

```
if (συνθήκη) {
  κώδικας προς εκτέλεση εάν η συνθήκη είναι αληθής;
} elseif (συνθήκη) {
  κώδικας προς εκτέλεση εάν η συνθήκη είναι αληθής;
```

```
} else {
  κώδικας προς εκτέλεση εάν δεν ισχύει καμία συνθήκη;}
```

Το παρακάτω παράδειγμα θα δώσει έξοδο: "Have a good morning!" εάν η τρέχουσα ώρα είναι μικρότερη από “10”, και “Have a good day!" εάν η τρέχουσα ώρα είναι λιγότερο από ‘20”. Διαφορετικά θα δώσει έξοδο "Have a good night!":

Παράδειγμα (php_28.php)

```
<?php
$t=date("H");

if ($t<"10") {
  echo "Have a good morning!";
} elseif ($t<"20") {
  echo "Have a good day!";
} else {
  echo "Have a good night!";
}
?>
```

PHP - Η εντολή switch

Η εντολή switch χρησιμοποιείται για να εκτελέσει διάφορες ενέργειες που βασίζονται σε διαφορετικές συνθήκες. Χρησιμοποιήστε την εντολή switch για να **επιλέξετε ένα από τα πολλά μπλοκ του κώδικα που πρόκειται να εκτελεστεί** .

Σύνταξη

```
switch (n) {
  case label1:
    κώδικας προς εκτέλεση αν  n=label1;
    break;
  case label2:
   κώδικας προς εκτέλεση αν  n=label2;
    break;
  case label3:
κώδικας προς εκτέλεση αν  n=label3;
    break;
  ...
  default:
```

κώδικας προς εκτέλεση αν n είναι διαφορετική από όλα τα παραπάνω labels;
}

Έτσι λοιπόν λειτουργεί: Πρώτα έχουμε μια ενιαία έκφραση *n* (πιο συχνά μια μεταβλητή), που αξιολογείται μία φορά. Η τιμή της έκφρασης στη συνέχεια συγκρίνεται με τις τιμές για κάθε περίπτωση στη δομή. Αν υπάρχει ένα ταίριασμα, το μπλοκ του κώδικα που σχετίζεται με την εν λόγω υπόθεση εκτελείται. Χρησιμοποιήστε το **break** για να αποτρέψετε την εκτέλεση του κώδικα στην επόμενη περίπτωση αυτόματα. Η **προεπιλεγμένη** δήλωση χρησιμοποιείται εάν δεν βρεθεί ταίριασμα.

Παράδειγμα (php_29.php)

```
<?php
$favcolor="red";

switch ($favcolor) {
  case "red":
    echo "Your favorite color is red!";
    break;
  case "blue":
    echo "Your favorite color is blue!";
    break;
  case "green":
    echo "Your favorite color is green!";
    break;
  default:
    echo "Your favorite color is neither red, blue, or green!";
}
?>
```

PHP Βρόχοι

Συχνά, όταν γράφετε κώδικα, θέλετε το ίδιο μπλοκ του κώδικα για να τρέξει ξανά και ξανά σε μια σειρά. Αντί της προσθήκης πολλών σχεδόν ίδιων γραμμών κώδικα σε ένα σενάριο, μπορούμε να χρησιμοποιούμε βρόχους για να εκτελέσουν ένα έργο όπως αυτό.

Στην PHP, έχουμε τις ακόλουθες εντολές επανάληψης:

- **while** - επανάληψη ενός μπλοκ του κώδικα όσο η καθορισμένη συνθήκη είναι αληθής
- **do ... while** – εκτέλεση ενός μπλοκ του κώδικα μια φορά, και στη συνέχεια επανάληψή του όσο η καθορισμένη συνθήκη είναι αληθής
- **for** - επανάληψη ενός μπλοκ του κώδικα για ένα συγκεκριμένο αριθμό φορών
- **foreach** - επανάληψη ενός μπλοκ του κώδικα για κάθε στοιχείο ενός πίνακα

PHP - while βρόχοι

Οι PHP while βρόχοι εκτελούν ένα μπλοκ του κώδικα όταν η καθορισμένη συνθήκη είναι αληθής.

Σύνταξη

while (*η συνθήκη είναι αληθής*) {
   κώδικας προς εκτέλεση;
}

Το παρακάτω παράδειγμα ορίζει μια πρώτη μεταβλητή $x ίση με 1 ($x = 1). Στη συνέχεια, ο βρόχος while θα συνεχίσει να λειτουργεί όσο το $x είναι μικρότερο ή ίσο με 5 ($x <= 5). Το $x θα αυξηθεί κατά 1 κάθε φορά που ο βρόχος εκτελείται ($x ++):

Παράδειγμα (php_30.php)

```
<?php
$x = 1;

while($x <= 5) {
   echo "The number is: $x <br>";
   $x++;
}
?>
```

Έξοδος:
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5

Ο PHP do ... while βρόχος

Ο do ... while βρόχος θα εκτελεί πάντα το μπλοκ του κώδικα μια φορά, θα ελέγχει στη συνέχεια την συνθήκη, και θα επαναλαμβάνει το βρόχο όσο η καθορισμένη συνθήκη είναι αληθής.

Σύνταξη

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

Το παρακάτω παράδειγμα ορίζει μια πρώτη μεταβλητή $x ίση με 1 ($x = 1). Στη συνέχεια, ο do while βρόχος θα γράψει κάποια έξοδο, και στη συνέχεια, αυξάνει τη μεταβλητή $x κατά 1. Στη συνέχεια, η συνθήκη ελέγχεται ($x είναι μικρότερη ή ίση με 5;), και ο βρόχος θα συνεχίσει να εκτελείται όσο $x είναι μικρότερο από ή ίσο με το 5:

Παράδειγμα (php_31.php)

```
<?php
$x = 1;

do {
   echo "The number is: $x <br>";
   $x++;
} while ($x <= 5);
?>
```

Η έξοδος και εδώ είναι όπως στο προηγούμενο παράδειγμα.

Σημειώστε ότι σε ένα do while βρόχο η συνθήκη ελέγχεται ΜΕΤΑ την εκτέλεση των δηλώσεων εντός του βρόχου. Αυτό σημαίνει ότι ο do while βρόχος θα εκτελέσει τις δηλώσεις του, τουλάχιστον μία φορά, ακόμη και αν η συνθήκη είναι ψευδής την πρώτη φορά.

Το παρακάτω παράδειγμα ορίζει τη μεταβλητή $x σε 6, έπειτα εκτελεί το βρόχο, **και στη συνέχεια η συνθήκη ελέγχεται**:

Παράδειγμα (php_32.php)

```
<?php
$x = 6;

do {
   echo "The number is: $x <br>";
   $x++;
} while ($x<=5);
?>
```

Έξοδος:
The number is: 6

PHP Ο βρόχος for…

Ο PHP βρόχος for… εκτελεί ένα μπλοκ του κώδικα ένα συγκεκριμένο αριθμό φορών.

Ο βρόχος χρησιμοποιείται όταν γνωρίζετε εκ των προτέρων πόσες φορές το σενάριο θα πρέπει να τρέξει.

Σύνταξη

for (*init counter; test counter; increment counter*) {
  κώδικας προς εκτέλεση;
}

Παράμετροι:

- *init counter*: Αρχικοποίηση της τιμής του μετρητή του βρόχου
- *test counter*: Αποτίμηση για κάθε επανάληψη βρόχου. Αν true, ο βρόχος συνεχίζει. Αν η τιμή είναι false, ο βρόχος τελειώνει.
- *increment counter*: Αυξάνει την τιμή του μετρητή βρόχου

Το παρακάτω παράδειγμα εμφανίζει τους αριθμούς από 0 έως και 10:

Παράδειγμα (php_33.php)

```
<!DOCTYPE html>
<html>
<body>

<?php
for ($x = 0; $x <= 10; $x++) {
  echo "The number is: $x <br>";
}
?>

</body>
</html>
```

Ο βρόχος foreach της PHP

Ο βρόχος foreach λειτουργεί μόνο σε πίνακες, και χρησιμοποιείται για να «σαρώνει» κάθε ζεύγος κλειδιού / τιμής σε ένα πίνακα.

Σύνταξη

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

Για κάθε επανάληψη βρόχου, η τιμή του τρέχοντος στοιχείου του πίνακα array εκχωρείται στη μεταβλητή value και ο δείκτης πίνακα κινείται κατά μια θέση, μέχρι να φτάσει στο τελευταίο στοιχείο του πίνακα.

Το παρακάτω παράδειγμα δείχνει ένα βρόχο που θα παράγει έξοδο τις τιμές του συγκεκριμένου πίνακα ($colors):

Παράδειγμα (php_34.php)

```
<!DOCTYPE html>
<html>
<body>

<?php
$colors = array("red", "green", "blue", "yellow");

foreach ($colors as $value) {
  echo "$value <br>";
}
?>

</body>
</html>
```

PHP - Συναρτήσεις

Η πραγματική δύναμη της PHP προέρχεται από τις συναρτήσεις της. Έχει περισσότερες από 1000 ενσωματωμένες συναρτήσεις.

PHP συναρτήσεις χρήστη

Εκτός από τις ενσωματωμένες συναρτήσεις PHP, μπορούμε να δημιουργήσουμε τις δικές μας συναρτήσεις. Μια συνάρτηση είναι ένα σύνολο εντολών που μπορεί να χρησιμοποιηθεί κατ' επανάληψη σε ένα πρόγραμμα. Μια συνάρτηση δεν θα εκτελεστεί αμέσως όταν φορτώσει μια σελίδα. Μια συνάρτηση θα εκτελεστεί μετά από κλήση που θα γίνει σε αυτήν.

Δημιουργήστε μια συνάρτηση οριζόμενη από το χρήστη σε PHP

Μια συνάρτηση οριζόμενη από το χρήστη ξεκινά με τη λέξη "function":

Σύνταξη

function *functionName*() {
  κωδικας προς εκτέλεση;
}

Σημείωση: Το όνομα της συνάρτησης μπορεί να αρχίζει με ένα γράμμα ή με υπογράμμιση (όχι με αριθμό).

Συμβουλή: Δώστε στη συνάρτηση ένα όνομα που να αντανακλά το τι κάνει η συνάρτηση!

 Τα ονόματα των συναρτήσεων ΔΕΝ είναι case-sensitive.

Στο παρακάτω παράδειγμα, δημιουργούμε μια συνάρτηση που ονομάζεται "writeMsg ()". Το άγκιστρο ανοίγματος ‘{‘ δείχνει την αρχή του κώδικα της συνάρτησης και το άγκιστρο κλείσιματος ‘}’ δείχνει το τέλος της συνάρτησης. Η συνάρτηση εξάγει "Hello world!". Για να καλέσετε τη συνάρτηση, απλά γράψτε το όνομά της:

Παράδειγμα (php_35.php)

```
<!DOCTYPE html>
<html>
<body>

<?php
function writeMsg() {
    echo "Hello world!";
}

writeMsg();
?>

</body>
</html>
```

Παράμετροι Συνάρτησης PHP

Πληροφορίες μπορούν να περάσουν στις συναρτήσεις μέσα από τις παραμέτρους. Μια παράμετρος είναι ακριβώς όπως μια μεταβλητή.

Οι παράμετροι καθορίζονται μετά από το όνομα της συνάρτησης, μέσα στις παρενθέσεις. Μπορείτε να προσθέσετε όσες παραμέτρους θέλετε, απλά τις χωρίζετε με ένα κόμμα.

Το ακόλουθο παράδειγμα έχει μια συνάρτηση με μια παράμετρο ($fname). Όταν η συνάρτηση familyName () καλείται, μπορούμε επίσης να περάσουμε μαζί ένα όνομα (π.χ. Jani), και το όνομα χρησιμοποιείται μέσα στη συνάρτηση η οποία εξάγει πολλά διαφορετικά ονόματα, αλλά ένα ίδιο επώνυμο:

Παράδειγμα (php_36.php)

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" >
</head>
<body>

<?php
function familyName($fname) {
    echo "$fname Refsnes.<br>";
}

familyName("Jani");
familyName("Hege");
familyName("Πέτρος");
familyName("Kai Jim");
familyName("Borge");
?>

</body>
</html>
```

Το ακόλουθο παράδειγμα έχει μια συνάρτηση με δύο παραμέτρους ($fname και $year):

Παράδειγμα (php_37.php)

```
<!DOCTYPE html>
<html>
<body>

<?php
function familyName($fname, $year) {
    echo "$fname Refsnes. Born in $year <br>";
}

familyName("Hege","1975");
familyName("Stale","1978");
familyName("Kai Jim","1983");
?>
```

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

PHP - Προεπιλεγμένη τιμή παραμέτρου

Το παρακάτω παράδειγμα δείχνει πώς να χρησιμοποιήσετε μια προκαθορισμένη παράμετρο. Εάν καλούμε την συνάρτηση setHeight() χωρίς παράμετρο, τότε παίρνει την προεπιλεγμένη τιμή ως παράμετρο:

Παράδειγμα (php_38.php)

```
<!DOCTYPE html>
<html>
<body>

<?php
function setHeight($minheight = 50) {
    echo "The height is : $minheight <br>";
}

setHeight(350);
setHeight();
setHeight(135);
setHeight(80);
?>

</body>
</html>
```

Συναρτήσεις PHP - Επιστρέφοντας τιμές

Για να αφήσετε μια συνάρτηση να επιστρέφει μια τιμή, χρησιμοποιήστε την εντολή return:

Παράδειγμα (php_39.php)

```
<!DOCTYPE html>
<html>
<body>

<?php
function sum($x, $y) {
    $z = $x + $y;
    return $z;
```

```
}

echo "5 + 10 = " . sum(5,10) . "<br>";
echo "7 + 13 = " . sum(7,13) . "<br>";
echo "2 + 4 = " . sum(2,4);
?>

</body>
</html>
```

PHP Πίνακες

Ένας πίνακας αποθηκεύει πολλαπλές τιμές σε μια μόνο μεταβλητή:

Παράδειγμα

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

Τι είναι ένας πίνακας;

Ένας πίνακας είναι μια ειδική μεταβλητή, η οποία μπορεί να κρατήσει περισσότερες από μία τιμή κάθε φορά.

Εάν έχετε μια λίστα με αντικείμενα (μια λίστα με τα ονόματα των αυτοκινήτων, για παράδειγμα), η αποθήκευση των αυτοκινήτων σε μεμονωμένες μεταβλητές θα μπορούσε να μοιάζει με αυτό:

```
$cars1 = "Volvo";
$cars2 = "BMW";
$cars3 = "Toyota";
```

Ωστόσο, τι συμβαίνει αν θέλετε να κάνετε loop μέσα από τα αυτοκίνητα και να βρείτε κάποιο συγκεκριμένο; Και τι θα συνέβαινε αν δεν είχατε 3 αυτοκίνητα, αλλά 300;

Η λύση είναι να δημιουργήσετε ένα πίνακα!

Ένας πίνακας μπορεί να κρατήσει πολλές τιμές κάτω από ένα όνομα, και μπορείτε να έχετε πρόσβαση στις τιμές με αναφορά σε έναν αριθμό δείκτη.

Δημιουργήστε ένα πίνακα στην PHP

Στην PHP, η συνάρτηση array () χρησιμοποιείται για να δημιουργήσει ένα πίνακα:

```
array();
```

Στην PHP, υπάρχουν τρεις τύποι των πινάκων:

- **Πίνακες με δείκτες** - Πίνακες με ένα αριθμητικό δείκτη
- **Συσχετιζόμενοι πίνακες** - Πίνακες με ονομαστικά κλειδιά
- **Πολυδιάστατοι πίνακες** - Πίνακες που περιέχουν έναν ή περισσότερους πίνακες

PHP πίνακες με δείκτες (indexed)

Υπάρχουν δύο τρόποι για να δημιουργήσετε πίνακες με δείκτες:

Ο δείκτης μπορεί να εκχωρηθεί αυτόματα (δείκτης ξεκινά πάντα από το 0), όπως αυτό:

```
$cars = array("Volvo", "BMW", "Toyota");
```

ή ο δείκτης μπορεί να εκχωρηθεί μη αυτόματα:

```
$cars[0] = "Volvo";
$cars[1] = "BMW";
$cars[2] = "Toyota";
```

Το ακόλουθο παράδειγμα δημιουργεί ένα ευρετήριο πίνακα που ονομάζεται $cars, εκχωρεί τρία στοιχεία σε αυτό, και στη συνέχεια εκτυπώνει ένα κείμενο που περιέχει τις τιμές πίνακα:

Παράδειγμα (php_40.php)

```
<!DOCTYPE html>
<html>
<body>

<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>

</body>
</html>
```

Πάρτε το μήκος ενός Array – Η συνάρτηση count()

Η συνάρτηση count() χρησιμοποιείται για να επιστρέψει το μήκος (τον αριθμό των στοιχείων) ενός πίνακα:

Παράδειγμα (php_41.php)

```
<!DOCTYPE html>
<html>
<body>

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

</body>
</html>
```

Βρόχος μέσω ενός δεικτοδοτούμενου πίνακα

Για να κάνετε loop και εκτύπωση σε όλες τις τιμές ενός δεικτοδοτούμενου πίνακα, θα μπορούσατε να χρησιμοποιήσετε ένα βρόχο for:

Παράδειγμα (php_42.php)

```
<!DOCTYPE html>
<html>
<body>

<?php
$cars = array("Volvo", "BMW", "Toyota");
$arrlength = count($cars);

for($x = 0; $x <  $arrlength; $x++) {
    echo $cars[$x];
    echo "<br>";
}
?>

</body>
</html>
```

PHP Συσχετιζόμενοι (associative) πίνακες

Είναι πίνακες που χρησιμοποιούν ονομαστικά κλειδιά που θα αποδώσετε σ' αυτούς.

Υπάρχουν δύο τρόποι για να δημιουργήσετε ένα συσχετιζόμενο πίνακα:

```
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
```

ή:

```
$age['Peter'] = "35";
$age['Ben'] = "37";
$age['Joe'] = "43";
```

Τα ονομαστικά κλειδιά μπορούν στη συνέχεια να χρησιμοποιηθούν σε ένα σενάριο:

Παράδειγμα (php_43.php)

```
<!DOCTYPE html>
<html>
<body>

<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old.";
?>

</body>
</html>
```

Loop Μέσω ενός Συσχετιζόμενου Πίνακα

Για να κάνετε loop και εκτύπωση όλων των τιμών από ένα συσχετιζόμενο πίνακα, μπορείτε να χρησιμοποιήσετε ένα βρόχο foreach:

Παράδειγμα (php_44.php)

```
<!DOCTYPE html>
<html>
<body>

<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

foreach($age as $x => $x_value) {
    echo "Key=" . $x . ", Value=" . $x_value;
```

```
    echo "<br>";
}
?>

</body>
</html>
```

Πολυδιάστατοι (multidimensional) πίνακες

Οι Πολυδιάστατοι πίνακες θα εξηγηθούν παρακάτω, στην ενότητα Advanced PHP.

Πλήρης PHP αναφορά πίνακα.

Για μια πλήρη αναφορά όλων των λειτουργιών του πίνακα, πηγαίνετε στη διεύθυνση http://www.w3schools.com/php/php_ref_array.asp.

Η αναφορά περιέχει μια σύντομη περιγραφή και παραδείγματα της χρήσης, για κάθε συνάρτηση!

PHP Ταξινόμηση Πινάκων

Τα στοιχεία σε έναν πίνακα μπορούν να ταξινομηθούν σε αλφαβητική ή αριθμητική σειρά, φθίνουσα ή αύξουσα σειρά.

PHP – Συναρτήσεις ταξινόμησης για τους Πίνακες

Σε αυτό το κεφάλαιο, θα περάσουμε στις ακόλουθες συναρτήσεις PHP ταξινόμησης πίνακα:

- sort() - ταξινόμηση πινάκων με αύξουσα σειρά
- rsort()- ταξινόμηση πινάκων με φθίνουσα σειρά
- asort() – ταξινόμηση συσχετιζόμενων πινάκων σε αύξουσα σειρά, σύμφωνα με την τιμή
- ksort() - ταξινόμηση συσχετιζόμενων πινάκων σε αύξουσα σειρά, σύμφωνα με το κλειδί

- arsort() - ταξινόμηση συσχετιζόμενων πινάκων σε φθίνουσα σειρά, σύμφωνα με την τιμή
- krsort() - ταξινόμηση συσχετιζόμενων πινάκων σε φθίνουσα σειρά, σύμφωνα με το κλειδί

Ταξινόμηση Array σε αύξουσα σειρά - sort()

Το ακόλουθο παράδειγμα ταξινομεί τα στοιχεία του πίνακα $cars με αύξουσα αλφαβητική σειρά:

Παράδειγμα (php_45.php)

```
<!DOCTYPE html>
<html>
<body>

<?php
$cars = array("Volvo", "BMW", "Toyota");
sort($cars);

$clength = count($cars);
for($x = 0; $x <  $clength; $x++) {
    echo $cars[$x];
    echo "<br>";
}
?>

</body>
</html>
```

Το ακόλουθο παράδειγμα ταξινομεί τα στοιχεία του πίνακα $numbers σε αύξουσα αριθμητική σειρά:

Παράδειγμα (php_46.php)

```
<!DOCTYPE html>
<html>
<body>

<?php
$numbers = array(4, 6, 2, 22, 11);
sort($numbers);

$arrlength = count($numbers);
for($x = 0; $x <  $arrlength; $x++) {
    echo $numbers[$x];
    echo "<br>";
```

```
}
?>

</body>
</html>
```

Ταξινόμηση Array σε φθίνουσα σειρά - rsort()

Το ακόλουθο παράδειγμα ταξινομεί τα στοιχεία του πίνακα $cars κατά φθίνουσα αλφαβητική σειρά:

Παράδειγμα (php_47.php)

```
<!DOCTYPE html>
<html>
<body>

<?php
$cars = array("Volvo", "BMW", "Toyota");
rsort($cars);

$clength = count($cars);
for($x = 0; $x <  $clength; $x++) {
    echo $cars[$x];
    echo "<br>";
}
?>

</body>
</html>
```

Το ακόλουθο παράδειγμα ταξινομεί τα στοιχεία του πίνακα $numbers σε φθίνουσα αριθμητική σειρά:

Παράδειγμα

```
<!DOCTYPE html>
<html>
<body>

<?php
$numbers = array(4, 6, 2, 22, 11);
rsort($numbers);

$arrlength = count($numbers);
for($x = 0; $x <  $arrlength; $x++) {
    echo $numbers[$x];
    echo "<br>";
}
```

```
?>

</body>
</html>
```

Ταξινόμηση Array σε αύξουσα σειρά, σύμφωνα με την τιμή - asort()

Το ακόλουθο παράδειγμα ταξινομεί ένα συσχετιζόμενο (associative) πίνακα με αύξουσα σειρά, σύμφωνα με την τιμή:

Παράδειγμα (php_48.php)

```
<!DOCTYPE html>
<html>
<body>

<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
asort($age);

foreach($age as $x => $x_value) {
    echo "Key=" . $x . ", Value=" . $x_value;
    echo "<br>";
}
?>

</body>
</html>
```

Έξοδος:
Key=Peter, Value=35
Key=Ben, Value=37
Key=Joe, Value=43

Ταξινόμηση Array σε αύξουσα σειρά, σύμφωνα με το κλειδί - ksort()

Το ακόλουθο παράδειγμα ταξινομεί ένα συσχετιζόμενο πίνακα με αύξουσα σειρά, σύμφωνα με το κλειδί:

Παράδειγμα

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

```
<body>

<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
ksort($age);

foreach($age as $x => $x_value) {
    echo "Key=" . $x . ", Value=" . $x_value;
    echo "<br>";
}
?>

</body>
</html>
```

Έξοδος:
Key=Ben, Value=37
Key=Joe, Value=43
Key=Peter, Value=35

Ταξινόμηση Array σε φθίνουσα σειρά, ανάλογα με την τιμή - arsort()

Το ακόλουθο παράδειγμα ταξινομεί ένα συσχετιζόμενο πίνακα κατά φθίνουσα σειρά, σύμφωνα με την τιμή:

Παράδειγμα (php_49.php)

```
<!DOCTYPE html>
<html>
<body>

<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
arsort($age);

foreach($age as $x => $x_value) {
    echo "Key=" . $x . ", Value=" . $x_value;
    echo "<br>";
}
?>

</body>
</html>
```

Έξοδος:
Key=Joe, Value=43

Key=Ben, Value=37
Key=Peter, Value=35

Ταξινόμηση Array σε φθίνουσα σειρά, σύμφωνα με το κλειδί - krsort()

Το ακόλουθο παράδειγμα ταξινομεί ένα συσχετιζόμενο πίνακα κατά φθίνουσα σειρά, σύμφωνα με το κλειδί:

Παράδειγμα (php_50.php)

```
<!DOCTYPE html>
<html>
<body>

<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
krsort($age);

foreach($age as $x => $x_value) {
    echo "Key=" . $x . ", Value=" . $x_value;
    echo "<br>";
}
?>

</body>
</html>
```

Έξοδος:
Key=Peter, Value=35
Key=Joe, Value=43
Key=Ben, Value=37

PHP Καθολικές μεταβλητές - superglobals

Οι superglobals εισήχθησαν στην PHP 4.1.0, και είναι ενσωματωμένες μεταβλητές που είναι πάντα διαθέσιμες σε όλα τα πεδία.

Πολλές προκαθορισμένες μεταβλητές στην PHP είναι «superglobals», πράγμα που σημαίνει ότι είναι πάντοτε διαθέσιμες, ανεξάρτητα από το πεδίο εφαρμογής - και μπορείτε να αποκτήσετε πρόσβαση σε αυτές από οποιαδήποτε συνάρτηση, κλάση ή αρχείο χωρίς να χρειάζεται να κάνετε τίποτα ιδιαίτερο.

Οι superglobals μεταβλητές της PHP είναι:

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

Το κεφάλαιο αυτό θα εξηγήσει ορισμένες επιλεγμένες μεταβλητές από τις superglobals.

PHP $GLOBALS

Η $GLOBALS είναι μια PHP σούπερ καθολική μεταβλητή που χρησιμοποιείται για πρόσβαση σε καθολικές μεταβλητές από οπουδήποτε στο σενάριο PHP (και μέσα από τις συναρτήσεις ή τις μεθόδους).

Η PHP αποθηκεύει όλες τις καθολικές μεταβλητές σε ένα πίνακα που ονομάζεται $GLOBALS[*index*]. Ο *δείκτης* διατηρεί το όνομα της μεταβλητής.

Το παρακάτω παράδειγμα δείχνει πώς μπορείτε να χρησιμοποιήσετε τη σούπερ καθολική μεταβλητή $GLOBALS:

Παράδειγμα (php_51.php)

```
<!DOCTYPE html>
<html>
<body>

<?php
```

```
$x = 75;
$y = 25;

function addition() {
    $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];
}

addition();
echo $z;
?>

</body>
</html>
```

Στο παραπάνω παράδειγμα, δεδομένου ότι το z είναι μια μεταβλητή εντός του πίνακα $GLOBALS, είναι επίσης προσβάσιμη και έξω από τη συνάρτηση!

PHP $ _SERVER

Η $ _SERVER είναι μια PHP σούπερ global μεταβλητή που περιέχει πληροφορίες σχετικά με τις κεφαλίδες, διαδρομές και το PHP σενάριο.

Το παρακάτω παράδειγμα δείχνει πώς να χρησιμοποιήσετε κάποια από τα στοιχεία του $_SERVER:

Παράδειγμα (php_52.php)

```
<!DOCTYPE html>
<html>
<body>

<?php
echo $_SERVER['PHP_SELF'];
echo "<br>";
echo $_SERVER['SERVER_NAME'];
echo "<br>";
echo $_SERVER['SERVER_SOFTWARE'];
echo "<br>";
echo $_SERVER['HTTP_HOST'];
echo "<br>";
echo $_SERVER['HTTP_REFERER'];
echo "<br>";
echo $_SERVER['HTTP_USER_AGENT'];
echo "<br>";
echo $_SERVER['SCRIPT_NAME'];
?>
```

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

Έξοδος:
```
/ASK_PHP_check/php_52.php
localhost
Apache/2.4.9 (Win32) OpenSSL/1.0.1g PHP/5.5.11
localhost
http://localhost/ASK_PHP_check/
Mozilla/5.0 (Windows NT 6.1; WOW64; rv:37.0) Gecko/20100101 Firefox/37.0
/ASK_PHP_check/php_52.php
```

Ο παρακάτω πίνακας παραθέτει τα πιο σημαντικά στοιχεία που μπορούν να πάνε μέσα στο $ _SERVER:

Element/Code	Description
$_SERVER['PHP_SELF']	Επιστρέφει το όνομα του script που εκτελείται αυτή τη στιγμή
$_SERVER['GATEWAY_INTERFACE']	Επιστρέφει την έκδοση του Common Gateway Interface (CGI) που χρησιμοποιεί ο διακομιστής
$_SERVER['SERVER_ADDR']	Επιστρέφει την διεύθυνση IP του κεντρικού διακομιστή
$_SERVER['SERVER_NAME']	Επιστρέφει το όνομα του κεντρικού διακομιστή (όπως www.w3schools.com)
$_SERVER['SERVER_SOFTWARE']	Επιστρέφει το αλφαριθμητικό ταυτότητας του διακομιστή (όπως Apache / 2.2.24)
$_SERVER['SERVER_PROTOCOL']	Επιστρέφει το όνομα και την αναθεώρηση του πρωτοκόλλου πληροφοριών (όπως το HTTP / 1.1)
$_SERVER['REQUEST_METHOD']	Επιστρέφει τη μέθοδο αίτημα που χρησιμοποιείται για πρόσβαση στη σελίδα (όπως POST)
$_SERVER['REQUEST_TIME']	Επιστρέφει τη χρονική σήμανση της έναρξης της αίτησης (όπως 1377687496)
$_SERVER['QUERY_STRING']	Επιστρέφει το string ερώτημα αν η σελίδα είναι προσβάσιμη μέσω μιας συμβολοσειράς ερωτήματος
$_SERVER['HTTP_ACCEPT']	Επιστρέφει την κεφαλίδα Accept από το τρέχον αίτημα
$_SERVER['HTTP_ACCEPT_CHARSET']	Επιστρέφει την κεφαλίδα Accept_Charset από την τρέχουσα αίτηση (όπως utf-8, ISO-8859-1)

$_SERVER['HTTP_HOST']	Επιστρέφει την κεφαλίδα κεντρικού υπολογιστή από το τρέχον αίτημα
$_SERVER['HTTP_REFERER']	Επιστρέφει την πλήρη διεύθυνση URL της τρέχουσας σελίδας (δεν είναι αξιόπιστη, επειδή δεν το υποστηρίζουν όλοι οι χρήστες-browsers)
$_SERVER['HTTPS']	Είναι το ερώτημα του σεναρίου μέσω ενός ασφαλούς πρωτοκόλλου HTTP
$_SERVER['REMOTE_ADDR']	Επιστρέφει τη διεύθυνση IP από όπου ο χρήστης βλέπει την τρέχουσα σελίδα
$_SERVER['REMOTE_HOST']	Επιστρέφει το όνομα του υπολογιστή από όπου ο χρήστης βλέπει την τρέχουσα σελίδα
$_SERVER['REMOTE_PORT']	Επιστρέφει την θύρα που χρησιμοποιείται στον υπολογιστή του χρήστη να επικοινωνεί με τον web server
$_SERVER['SCRIPT_FILENAME']	Επιστρέφει την απόλυτη διαδρομή του σεναρίου που εκτελείται αυτή τη στιγμή
$_SERVER['SERVER_ADMIN']	Επιστρέφει την τιμή που δίνεται στην οδηγία SERVER_ADMIN στο αρχείο ρυθμίσεων του web server (αν το σενάριό σας τρέχει σε μια εικονική μηχανή, θα είναι η τιμή που ορίζεται για την εν λόγω εικονική μηχανή) (όπως someone@w3schools.com)
$_SERVER['SERVER_PORT']	Επιστρέφει στην θύρα με το μηχάνημα διακομιστή που χρησιμοποιείται από τον web server για την επικοινωνία (όπως 80)
$_SERVER['SERVER_SIGNATURE']	Επιστρέφει την έκδοση του διακομιστή και το εικονικό όνομα κεντρικού υπολογιστή που προστίθενται σε σελίδες που δημιουργούνται από τον εξυπηρετητή
$_SERVER['PATH_TRANSLATED']	Επιστρέφει τη διαδρομή που βασίζεται στο σύστημα αρχείων του τρέχοντος script
$_SERVER['SCRIPT_NAME']	Επιστρέφει τη διαδρομή του τρέχοντος σεναρίου
$_SERVER['SCRIPT_URI']	Επιστρέφει το URL της τρέχουσας σελίδας

PHP $ _REQUEST

Η PHP $ _REQUEST χρησιμοποιείται για τη συλλογή των δεδομένων μετά την υποβολή μιας φόρμας HTML.

Το παρακάτω παράδειγμα δείχνει μια φόρμα με ένα πεδίο εισαγωγής και ένα κουμπί υποβολής. Όταν ένας χρήστης υποβάλλει τα στοιχεία κάνοντας κλικ στο "Υποβολή" (Submit), τα δεδομένα της φόρμας στέλνονται στο αρχείο που ορίζεται στο χαρακτηριστικό ενέργειας (action) της ετικέτας <form>. Σε αυτό το παράδειγμα, δείχνουμε αυτό το ίδιο αρχείο PHP για επεξεργασία των δεδομένων της φόρμας. Εάν επιθυμείτε να χρησιμοποιήσετε ένα άλλο αρχείο PHP για την επεξεργασία δεδομένων φόρμας, αντικαταστήστε το με το όνομα της επιλογής σας. Στη συνέχεια, μπορούμε να χρησιμοποιήσουμε τη σούπερ καθολική μεταβλητή $ _REQUEST για να συλλέξουμε την τιμή του πεδίου εισαγωγής:

Παράδειγμα (php_53.php)

```
<!DOCTYPE html>
<html>
<body>

<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // collect value of input field
    $name = $_REQUEST['fname'];
    if (empty($name)) {
        echo "Name is empty";
    } else {
        echo $name;
    }
}
?>

</body>
</html>
```

PHP $ _POST

Η PHP $ _POST χρησιμοποιείται ευρέως για τη συλλογή των δεδομένων μετά την υποβολή μιας φόρμας HTML με τη μέθοδο "post". Η $ _POST χρησιμοποιείται επίσης ευρέως για να περάσει μεταβλητές.

Το παρακάτω παράδειγμα δείχνει μια φόρμα με ένα πεδίο εισαγωγής και ένα κουμπί υποβολής. Όταν ένας χρήστης υποβάλλει τα στοιχεία κάνοντας κλικ στο "Αποστολή", τα δεδομένα της φόρμας στέλνονται στο αρχείο που ορίζεται στο χαρακτηριστικό ενέργειας της ετικέτας <form>. Σε αυτό το παράδειγμα, έχουμε επισημάνει στο ίδιο το αρχείο για επεξεργασία δεδομένων φόρμας. Εάν επιθυμείτε να χρησιμοποιήσετε ένα άλλο αρχείο PHP για την επεξεργασία δεδομένων φόρμας, αντικαταστήστε το με το όνομα της επιλογής σας. Στη συνέχεια, μπορούμε να χρησιμοποιήσουμε τη σούπερ καθολική μεταβλητή $ _POST για να συλλέξουμε την τιμή του πεδίου εισαγωγής:

Παράδειγμα (php_54.php)

```
<!DOCTYPE html>
<html>
<body>

<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // collect value of input field
    $name = $_POST['fname'];
    if (empty($name)) {
        echo "Name is empty";
    } else {
        echo $name;
    }
}
?>

</body>
</html>
```

PHP $ _GET

Η PHP $ _GET μπορεί επίσης να χρησιμοποιηθεί για τη συλλογή στοιχείων μετά την υποβολή μιας φόρμας HTML με τη μέθοδο "get".

Η $ _GET μπορεί επίσης να συλλέξει δεδομένα που αποστέλλονται στη διεύθυνση URL.

Ας υποθέσουμε ότι έχουμε μια σελίδα HTML που περιέχει μια υπερ-σύνδεση με παραμέτρους:

```
<html>
<body>

<a href="test_get.php?subject=PHP&web=W3schools.com">Test $GET</a>

</body>
</html>
```

Όταν ένας χρήστης κάνει κλικ στο σύνδεσμο "Test $GET", οι παράμετροι "subject" και "web" έχουν αποσταλεί στο «test_get.php», και στη συνέχεια μπορείτε να έχετε πρόσβαση στις τιμές τους στο "test_get.php" με την $ _GET.

Παράδειγμα (php_55.php)

```
<!DOCTYPE html>
<html>
<body>

<a href="test_get.php?subject=PHP&web=W3schools.com">Test $GET</a>

</body>
</html>
```

Το παρακάτω παράδειγμα δείχνει τον κώδικα στο "test_get.php":

Παράδειγμα (test_get.php)

```
<html>
<body>

<?php
echo "Study " . $_GET['subject'] . " at " . $_GET['web'];
?>

</body>
</html>
```

Έξοδος:
Study PHP at W3schools.com

Συμβουλή: Θα μάθετε περισσότερα για τις $ _POST και $ _GET στο επόμενο κεφάλαιο για τις φόρμες PHP.

PHP Forms

PHP Χειρισμός φόρμας

Όπως είδαμε σε κάποια παραδείγματα, οι PHP superglobals $ _GET και $ _POST χρησιμοποιούνται για τη συλλογή δεδομένων φόρμας.

PHP - μια απλή φόρμα HTML

Το παρακάτω παράδειγμα εμφανίζει μια απλή HTML φόρμα με δύο πεδία εισόδου και ένα κουμπί υποβολής:

Παράδειγμα (php_56.php)

```
<!DOCTYPE HTML>
<html>
<body>

<form action="welcome.php" method="post">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

Όταν ο χρήστης συμπληρώνει την παραπάνω φόρμα και κάνει κλικ στο κουμπί υποβολής, τα δεδομένα φόρμας αποστέλλονται για επεξεργασία σε ένα αρχείο PHP που ονομάζεται "welcome.php". Τα δεδομένα φόρμας αποστέλλονται με τη μέθοδο HTTP POST.

Για να εμφανίσετε τα δεδομένα που υποβλήθηκαν θα μπορούσατε απλά να επαναλάβετε όλες τις μεταβλητές. Το "welcome.php" μοιάζει με αυτό:

Παράδειγμα (welcome.php)

```
<html>
<body>

Welcome <?php echo $_POST["name"]; ?><br>
Your email address is: <?php echo $_POST["email"]; ?>
```

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

Η έξοδος θα μπορούσε να είναι κάτι σαν αυτό:

```
Welcome John
Your email address is john.doe@example.com
```

Το ίδιο αποτέλεσμα θα μπορούσε να επιτευχθεί χρησιμοποιώντας τη μέθοδο HTTP GET:

Παράδειγμα

```
<html>
<body>

<form action="welcome_get.php" method="get">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

και το "welcome_get.php" μοιάζει με αυτό:

```
<html>
<body>

Welcome <?php echo $_GET["name"]; ?><br>
Your email address is: <?php echo $_GET["email"]; ?>

</body>
</html>
```

Ο κώδικας παραπάνω είναι αρκετά απλός. Ωστόσο, το πιο σημαντικό πράγμα λείπει. Θα πρέπει να επικυρώσετε τα δεδομένα φόρμας για να προστατεύσετε το σενάριό σας από κακόβουλο κώδικα.

Σκεφτείτε την ΑΣΦΑΛΕΙΑ κατά την επεξεργασία των φορμών PHP!

Αυτή η σελίδα δεν περιέχει καμία επικύρωση φόρμας, δείχνει ακριβώς πώς μπορείτε να στέλνετε και να ανακτήσετε τα δεδομένα φόρμας.

Ωστόσο, οι επόμενες σελίδες θα σας δείξουν πώς να επεξεργαστείτε τις φόρμες PHP με ασφάλεια! Η σωστή επικύρωση των δεδομένων φόρμας είναι σημαντικό για να προστατεύσει τη φόρμα σας από τους χάκερς και spammers!

GET εναντίον POST

Τόσο η GET όσο και η POST δημιουργούν ένα πίνακα (π.χ. πίνακα (key => value, key2 => value2, key3 => value3, ...)). Αυτός ο πίνακας κατέχει ζεύγη keys / values, όπου τα keys είναι τα ονόματα των στοιχείων ελέγχου φόρμας και values είναι τα δεδομένα εισόδου από το χρήστη.

Τόσο η GET όσο και η POST αντιμετωπίζονται ως $ _GET και $ _POST. Αυτές είναι superglobals, πράγμα που σημαίνει ότι είναι πάντοτε διαθέσιμες, ανεξάρτητα από το πεδίο εφαρμογής - και μπορείτε να αποκτήσετε πρόσβαση σε αυτές από οποιαδήποτε συνάρτηση, κλάση ή αρχείο χωρίς να χρειάζεται να κάνετε τίποτα ιδιαίτερο.

$ _GET Είναι ένας πίνακας από μεταβλητές που περνάνε στο τρέχον script μέσω των παραμέτρων URL.

$ _POST Είναι ένας πίνακας από μεταβλητές που περνάνε στο τρέχον script με τη μέθοδο HTTP POST.

Πότε να χρησιμοποιήσετε GET;

Οι πληροφορίες που στέλνονται από μια φόρμα με τη μέθοδο GET είναι **ορατές σε όλους** (όλα τα ονόματα των μεταβλητών και οι τιμές τους εμφανίζονται στο URL). Η GET έχει επίσης όρια στην ποσότητα των πληροφοριών για την αποστολή. Ο περιορισμός είναι περίπου 2000 χαρακτήρες. Ωστόσο, επειδή οι μεταβλητές εμφανίζονται στη διεύθυνση URL, είναι δυνατόν να θέσετε τη σελίδα ως σελιδοδείκτη. Αυτό μπορεί να είναι χρήσιμο σε ορισμένες περιπτώσεις.

Η GET μπορεί να χρησιμοποιηθεί για την αποστολή μη ευαίσθητων δεδομένων.

Σημείωση: Η GET δεν πρέπει ποτέ να χρησιμοποιείται για την αποστολή κωδικών πρόσβασης ή άλλων ευαίσθητων πληροφοριών!

Πότε να χρησιμοποιήσετε POST;

Οι πληροφορίες που στέλνονται από μια φόρμα με τη μέθοδο POST είναι **αόρατες στους άλλους** (όλα τα ονόματα / οι τιμές είναι ενσωματωμένα μέσα στο σώμα του αιτήματος HTTP) και δεν έχει **κανένα όριο** στην ποσότητα των πληροφοριών για την αποστολή.

Επιπλέον η POST υποστηρίζει προηγμένες λειτουργικότητα, όπως υποστήριξη για πολυμερή (multi-part) δυαδική είσοδο κατά το ανέβασμα αρχείων στον server.

Ωστόσο, επειδή οι μεταβλητές δεν εμφανίζονται στη διεύθυνση URL, δεν είναι δυνατόν να προσθέσετε τη σελίδα στους σελιδοδείκτες.

 **Οι προγραμματιστές προτιμούν POST για την αποστολή δεδομένων φόρμας.**

Επόμενο; ας δούμε πώς μπορούμε να επεξεργαστούμε τις φόρμες PHP με τον ασφαλή τρόπο!

PHP Εγκυρότητα φόρμας (Form Validation)

Αυτό και τα επόμενα κεφάλαια δείχνουν πώς να χρησιμοποιήσετε το PHP για την επικύρωση των δεδομένων φόρμας.

PHP Εγκυρότητα Φόρμας

Σκεφτείτε την ΑΣΦΑΛΕΙΑ κατά την επεξεργασία των φορμών PHP!

Η φόρμα HTML που θα δουλεύουμε σε αυτά τα κεφάλαια, περιέχει διάφορα πεδία εισαγωγής: απαιτούνται και προαιρετικά πεδία κειμένου, κουμπιά επιλογής, και ένα κουμπί υποβολής.

Οι κανόνες επικύρωσης για την παρακάτω φόρμα είναι οι εξής:

Πεδίο	Κανόνες εγκυρότητας
Name	Υποχρεωτικό. + Πρέπει να περιέχει μόνο γράμματα και κενό διάστημα
E-mail	Υποχρεωτικό. + Πρέπει να περιέχει μια έγκυρη email διεύθυνση (με @ και .)
Website	Προαιρετικό. Αν υπάρχει, περιλαμβάνει ένα έγκυρο URL
Comment	Προαιρετικό. Κείμενο εισόδου πολλαπλών γραμμών (textarea)
Gender	Υποχρεωτικό. Πρέπει να επιλέξετε ένα

PHP Form Validation Example

* required field.

Name: *

E-mail: *

Website:

Comment:

Gender: Female Male *

Submit

Your Input:

Πρώτα θα δούμε τον απλό κώδικα HTML για τη φόρμα:

Πεδία κειμένου

Το όνομα, e-mail, και website είναι στοιχεία εισαγωγής κειμένου (text input), και το πεδίο του σχόλιου είναι μια περιοχή κειμένου (textarea). Ο κώδικας HTMLλοιπόν:

```
Name: <input type="text" name="name">
E-mail: <input type="text" name="email">
Website: <input type="text" name="website">
Comment: <textarea name="comment" rows="5" cols="40"></textarea>
```

Κουμπιά επιλογής (radio buttons)

Τα πεδία για το φύλο (gender) είναι κουμπιά και ο κώδικας HTML μοιάζει με αυτό:

Gender:
<input type="radio" name="gender" value="female">Female
<input type="radio" name="gender" value="male">Male

Το στοιχείο φόρμας (form element)

Ο κώδικας HTML της φόρμας:

<form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">

Κατά την υποβολή της φόρμας, τα δεδομένα φόρμας αποστέλλονται με τη μέθοδο "post".

Ποια είναι η μεταβλητή $ _SERVER ["PHP_SELF"];

Η $ _SERVER ["PHP_SELF"] είναι μια super καθολική μεταβλητή που επιστρέφει το όνομα του script που εκτελείται κάθε φορά.

Έτσι, η $ _SERVER ["PHP_SELF"] στέλνει τα υποβληθέντα δεδομένα της φόρμας στην ίδια τη σελίδα, αντί να πηδά σε μια διαφορετική σελίδα. Με αυτό τον τρόπο, ο χρήστης θα πάρει τα μηνύματα λάθους στην ίδια σελίδα με τη φόρμα.

Τι είναι η htmlspecialchars () συνάρτηση;

Η htmlspecialchars () μετατρέπει ειδικούς χαρακτήρες σε HTML οντότητες. Αυτό σημαίνει ότι θα αντικαταστήσει χαρακτήρες HTML, όπως '<' και '>' με < και >. Αυτό αποτρέπει τους εισβολείς από την εκμετάλλευση του κώδικα με «ένεση» κώδικα HTML ή Javascript (επιθέσεις cross-site scripting) σε φόρμες.

Σημαντική σημείωση στην Ασφάλεια PHP Φόρμας

Η $ _SERVER ["PHP_SELF"] μεταβλητή μπορεί να χρησιμοποιηθεί από τους χάκερς!

Αν η PHP_SELF χρησιμοποιείται στη σελίδα σας, τότε ο χρήστης μπορεί να εισάγει μια κάθετο (/), και στη συνέχεια κάποιες Cross-Site Scripting (XSS) εντολές για να τις εκτελέσει.

Cross-site Scripting (XSS) είναι ένα είδος ευπάθειας ασφαλείας υπολογιστών που συναντάμε συνήθως σε εφαρμογές Web. Το XSS επιτρέπει στους επιτιθέμενους να εισάγουν client-side script σε ιστοσελίδες που προβάλλονται σε άλλους χρήστες.

Ας υποθέσουμε ότι έχουμε την παρακάτω φόρμα σε μια σελίδα με το όνομα "test_form.php":

```
<form method="post" action="<?php echo $_SERVER["PHP_SELF"];?>">
```

Τώρα, εάν ένας χρήστης εισάγει την κανονική διεύθυνση URL στη γραμμή διευθύνσεων, όπως "http://www.example.com/test_form.php", ο παραπάνω κώδικας θα μεταφραστεί σε:

```
<form method="post" action="test_form.php">
```

Μέχρι στιγμής, όλα καλά.

Ωστόσο, θεωρείστε ότι ένας χρήστης εισάγει την ακόλουθη διεύθυνση URL στη γραμμή διευθύνσεων:

```
http://www.example.com/test_form.php/%22%3E%3Cscript%3Ealert('hacked')%3C/script%3E
```

Στην περίπτωση αυτή, ο ανωτέρω κώδικας θα μεταφραστεί σε:

```
<form method="post" action="test_form.php/"><script>alert('hacked')</script>
```

Αυτός ο κωδικός προσθέτει μια ετικέτα script και μια εντολή συναγερμού. Και όταν η σελίδα φορτώσει, θα εκτελεστεί ο κώδικας JavaScript (ο χρήστης θα δει ένα πλαίσιο ειδοποίησης). Αυτό είναι μόνο ένα απλό και ακίνδυνο παράδειγμα πώς μπορεί να αξιοποιηθεί η μεταβλητή PHP_SELF.

ΔΟΚΙΜΑΣΤΕ ΤΟ: το προηγούμενο παράδειγμα php_53.php κάνει χρήση της PHP_SELF. Οπότε, δοκιμάστε στο URL του browser αντί για «…/php_53.php» να δώσετε «… /php_53.php/%22%3E%3Cscript%3Ealert('hacked')%3C/script%3E»

Να θυμάστε ότι **κάθε κώδικας JavaScript μπορεί να προστεθεί μέσα στη <script> ετικέτα!** Ένας χάκερ μπορεί να ανακατευθύνει τον χρήστη σε ένα αρχείο σε άλλο διακομιστή, και αυτό το αρχείο μπορεί να περιέχει κακόβουλο κώδικα που μπορεί να μεταβάλει τις καθολικές μεταβλητές ή να υποβάλει τη φόρμα σε άλλη διεύθυνση για να αποθηκεύσει τα δεδομένα του χρήστη, για παράδειγμα.

Πώς να αποφύγετε τις $ _SERVER ["PHP_SELF"] επιθέσεις (exploits);

Τα $ _SERVER ["PHP_SELF"] exploits μπορεί να αποφευχθούν με τη χρήση της htmlspecialchars () συνάρτησης.

Ο κώδικας φόρμας θα πρέπει να μοιάζει κάπως έτσι:

<form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">

Η συνάρτηση htmlspecialchars () μετατρέπει ειδικούς χαρακτήρες σε HTML οντότητες. Τώρα, αν ο χρήστης προσπαθήσει να εκμεταλλευτεί την μεταβλητή PHP_SELF, αυτό θα οδηγήσει στο ακόλουθο αποτέλεσμα:

<form method="post"
action="test_form.php/"><script>alert('hacked')</script>">

Η προσπάθεια exploit αποτυγχάνει, και καμία ζημιά δεν λαμβάνει χώρα!

Παράδειγμα (php_53sec.php)

```
<!DOCTYPE html>
<html>
<body>

<form method="post" action="<?php echo htmlspecialchars
($_SERVER['PHP_SELF']);?>">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
   // collect value of input field
   $name = $_REQUEST['fname'];
   if (empty($name)) {
      echo "Name is empty";
   } else {
      echo $name;
   }
}
?>
```

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

Εγκυρότητα δεδομένων φόρμας με την PHP

Το πρώτο πράγμα που θα κάνουμε είναι να περάσουμε όλες τις μεταβλητές μέσω της PHP συνάρτησης htmlspecialchars().

Όταν χρησιμοποιούμε την htmlspecialchars () συνάρτηση, τότε αν ένας χρήστης προσπαθήσει να υποβάλει τα ακόλουθα σε ένα πεδίο κειμένου:

<script> location.href (‘http://www.hacked.com’) </script>

- Αυτό δεν θα εκτελεστεί, επειδή θα αποθηκευτεί ως HTML escaped κώδικας, όπως αυτό:

<script>location.href (‘http://www.hacked.com')</script>

Ο κώδικας είναι πλέον ασφαλές να εμφανίζεται σε μια σελίδα ή μέσα σε ένα e-mail.

Επίσης, θα κάνουμε ακόμα δύο πράγματα όταν ο χρήστης υποβάλλει τη φόρμα:

1. Αφαιρέστε περιττούς χαρακτήρες (space, tab, newline) από τα δεδομένα που εισάγει ο χρήστης (με την PHP συνάρτηση trim())
2. Αφαιρέστε backslashes (\) από τα δεδομένα που εισάγει ο χρήστης (με τη PHP συνάρτηση stripslashes())

Το επόμενο βήμα είναι να δημιουργήσουμε μια συνάρτηση που θα κάνει όλο τον έλεγχο για εμάς (το οποίο είναι πολύ πιο εύκολο από ό,τι γράφοντας τον ίδιο κώδικα ξανά και ξανά).

Θα ονομάσουμε τη συνάρτηση test_input().

```
<?php
// define variables and set to empty values
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $name = test_input($_POST["name"]);
  $email = test_input($_POST["email"]);
  $website = test_input($_POST["website"]);
  $comment = test_input($_POST["comment"]);
  $gender = test_input($_POST["gender"]);
}
```

```
function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>
```

Τώρα, μπορούμε να ελέγξουμε κάθε $ _POST μεταβλητή με τη συνάρτηση test_input(), και το σενάριο μοιάζει με αυτό:

Παράδειγμα (php_57.php)

```
<!DOCTYPE HTML>
<html>
<head>
</head>
<body>

<?php
// define variables and set to empty values
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
   $name = test_input($_POST["name"]);
   $email = test_input($_POST["email"]);
   $website = test_input($_POST["website"]);
   $comment = test_input($_POST["comment"]);
   $gender = test_input($_POST["gender"]);
}

function test_input($data) {
   $data = trim($data);
   $data = stripslashes($data);
   $data = htmlspecialchars($data);
   return $data;
}
?>

<h2>PHP Form Validation Example</h2>
<form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">
   Name: <input type="text" name="name">
   <br><br>
   E-mail: <input type="text" name="email">
   <br><br>
   Website: <input type="text" name="website">
   <br><br>
   Comment: <textarea name="comment" rows="5" cols="40"></textarea>
   <br><br>
```

```
  Gender:
  <input type="radio" name="gender" value="female">Female
  <input type="radio" name="gender" value="male">Male
  <br><br>
  <input type="submit" name="submit" value="Submit">
</form>

<?php
echo "<h2>Your Input:</h2>";
echo $name;
echo "<br>";
echo $email;
echo "<br>";
echo $website;
echo "<br>";
echo $comment;
echo "<br>";
echo $gender;
?>

</body>
</html>
```

Παρατηρήστε ότι στην αρχή του σεναρίου, ελέγχουμε αν η φόρμα έχει υποβληθεί χρησιμοποιώντας $ _SERVER ["REQUEST_METHOD"]. Εάν η REQUEST_METHOD είναι POST, τότε η φόρμα έχει υποβληθεί - και πρέπει να είναι επικυρωμένη. Αν δεν έχει υποβληθεί, παραλείψτε την επικύρωση και εμφανίστε μια κενή φόρμα.

Ωστόσο, στο παραπάνω παράδειγμα, όλα τα πεδία εισόδου είναι προαιρετικά. Το σενάριο δουλεύει μια χαρά ακόμα και αν ο χρήστης δεν εισάγει όλα τα δεδομένα.

Το επόμενο βήμα είναι να κάνουμε υποχρεωτικά τα πεδία εισαγωγής και να δημιουργήσουμε μηνύματα λάθους, αν χρειαστεί.

PHP Φόρμες-Υποχρεωτικά πεδία (form required)

Αυτό το κεφάλαιο δείχνει πώς να κάνετε τα πεδία εισαγωγής υποχρεωτικά, και να δημιουργήσετε μηνύματα λάθους, αν χρειαστεί.

PHP - Υποχρεωτικά πεδία

Από τον πίνακα κανόνων επικύρωσης, βλέπουμε ότι το "Όνομα", "E-mail", και "Φύλο" είναι πεδία υποχρεωτικά. Αυτά τα πεδία δεν μπορεί να είναι κενά και θα πρέπει να συμπληρωθούν στη μορφή HTML.

Πεδίο	Κανόνες επικύρωσης
Name	Υποχρεωτικό. + Πρέπει να περιέχει μόνο γράμματα και κενά διαστήματα
E-mail	Υποχρεωτικό. + Πρέπει να περιέχει μια έγκυρη email διεύθυνση (με @ και .)
Website	Προαιρετικό. Αν συμπληρωθεί, πρέπει να περιέχει ένα έγκυρο URL
Comment	Προαιρετικό. Πεδίο εισαγωγής με δυνατότητα συμπλήρωσης κειμένου σε πολλαπλές γραμμές (textarea)
Gender	Υποχρεωτικό. Αποκλειστική επιλογή ενός υποχρεωτικά

Στο προηγούμενο παράδειγμα, όλα τα πεδία εισαγωγής ήταν προαιρετικά. Στον κώδικα που ακολουθεί, έχουμε προσθέσει κάποια νέες μεταβλητές: $nameErr, $emailErr, $genderErr, και $websiteErr. Αυτές οι μεταβλητές σφάλματος θα κρατήσουν τα μηνύματα λάθους για τα υποχρεωτικά πεδία. Έχουμε προσθέσει επίσης και μια if else δήλωση για κάθε μεταβλητή $ _POST. Αυτή ελέγχει αν η μεταβλητή $ _POST είναι κενή (με την PHP συνάρτηση empty ()). Εάν είναι κενή, ένα μήνυμα σφάλματος αποθηκεύεται στις διάφορες μεταβλητές σφάλματος, και αν δεν είναι κενή, αποστέλλει τα δεδομένα εισόδου χρήστη μέσω της συνάρτησης test_input():

```
<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["name"])) {
    $nameErr = "Name is required";
  } else {
    $name = test_input($_POST["name"]);
  }

  if (empty($_POST["email"])) {
    $emailErr = "Email is required";
  } else {
    $email = test_input($_POST["email"]);
  }
```

```
  if (empty($_POST["website"])) {
    $website = "";
  } else {
    $website = test_input($_POST["website"]);  }

  if (empty($_POST["comment"])) {
    $comment = "";
  } else {
    $comment = test_input($_POST["comment"]);  }

  if (empty($_POST["gender"])) {
    $genderErr = "Gender is required";
  } else {
    $gender = test_input($_POST["gender"]);  }
}
?>
```

PHP - Εμφάνιση των μηνυμάτων λάθους

Στη συνέχεια, με τη μορφή HTML, προσθέτουμε ένα μικρό σενάριο (script) μετά από κάθε απαιτούμενο πεδίο, το οποίο δημιουργεί το σωστό μήνυμα λάθους, εάν χρειάζεται (δηλαδή αν ο χρήστης επιχειρεί να υποβάλει τη φόρμα χωρίς να συμπληρώσει τα υποχρεωτικά πεδία):

Ενδεικτικά, μετά από το <input type="text" name="name">, ένα script

```
<?php echo $nameErr;?>
```

Παράδειγμα (php_58.php)

```
<!DOCTYPE HTML>
<html>
<head>
<style>
.error {color: #FF0000;}
</style>
</head>
<body>

<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
   if (empty($_POST["name"])) {
     $nameErr = "Name is required";
```

```
  } else {
    $name = test_input($_POST["name"]);
  }

  if (empty($_POST["email"])) {
    $emailErr = "Email is required";
  } else {
    $email = test_input($_POST["email"]);   }

  if (empty($_POST["website"])) {
    $website = "";
  } else {
    $website = test_input($_POST["website"]);   }

  if (empty($_POST["comment"])) {
    $comment = "";
  } else {
    $comment = test_input($_POST["comment"]);   }

  if (empty($_POST["gender"])) {
    $genderErr = "Gender is required";
  } else {
    $gender = test_input($_POST["gender"]);
  }
}

function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>

<h2>PHP Form Validation Example</h2>
<p><span class="error">* required field.</span></p>
<form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">
  Name: <input type="text" name="name">
  <span class="error">* <?php echo $nameErr;?></span>
  <br><br>
  E-mail: <input type="text" name="email">
  <span class="error">* <?php echo $emailErr;?></span>
  <br><br>
  Website: <input type="text" name="website">
  <span class="error"><?php echo $websiteErr;?></span>
  <br><br>
  Comment: <textarea name="comment" rows="5" cols="40"></textarea>
  <br><br>
  Gender:
  <input type="radio" name="gender" value="female">Female
  <input type="radio" name="gender" value="male">Male
  <span class="error">* <?php echo $genderErr;?></span>
```

```
  <br><br>
  <input type="submit" name="submit" value="Submit">
</form>

<?php
echo "<h2>Your Input:</h2>";
echo $name;
echo "<br>";
echo $email;
echo "<br>";
echo $website;
echo "<br>";
echo $comment;
echo "<br>";
echo $gender;
?>

</body>
</html>
```

Το επόμενο βήμα είναι η επικύρωση των δεδομένων εισόδου, που είναι "Μήπως το όνομα του πεδίου περιέχει μόνο γράμματα και κενά;", και "Μήπως το πεδίο E-mail περιέχει μια έγκυρη σύνταξη διεύθυνση e-mail;», και αν είναι συμπληρωμένο, "Μήπως το πεδίο Ιστοσελίδα περιέχει μια έγκυρη διεύθυνση URL;»

PHP Φόρμες - Επικύρωση e-mail και URL

Αυτό το κεφάλαιο δείχνει τον τρόπο για να επικυρώσετε ονόματα, e-mails, και διευθύνσεις URL.

PHP - Επικύρωση Ονόματος

Ο παρακάτω κώδικας δείχνει έναν απλό τρόπο για να ελέγξετε αν το πεδίο όνομα περιέχει μόνο γράμματα και κενά. Εάν η τιμή του πεδίου ονόματος δεν είναι έγκυρη, τότε αποθηκεύστε ένα μήνυμα λάθους:

```
$name = test_input($_POST["name"]);
if (!preg_match("/^[a-zA-Z ]*$/",$name)) {
  $nameErr = "Only letters and white space allowed";
}
```

Η συνάρτηση preg_match() αναζητά μια συμβολοσειρά για το μοτίβο, επιστρέφοντας true αν υπάρχει το μοτίβο, και false αλλιώς.

Στη διεύθυνση http://www.phpro.org/tutorials/Introduction-to-PHP-Regex.html **μπορείτε να βρείτε λεπτομέρειες για το πώς κατασκευάζουμε τη πρώτη παράμετρο pattern της preg_match(), δηλαδή γενικότερα πως χειριζόμαστε regular expressions στην PHP.**

PHP - Επικύρωση E-mail

Ο ευκολότερος και ασφαλέστερος τρόπος για να ελέγξετε αν μια διεύθυνση ηλεκτρονικού ταχυδρομείου είναι γραμμένη σε σωστή και έγκυρη μορφή είναι να χρησιμοποιήσετε τη συνάρτηση της PHP filter_var().

Στον παρακάτω κώδικα, εάν η διεύθυνση ηλεκτρονικού ταχυδρομείου δεν είναι καλοσχηματισμένη, τότε αποθηκεύεται ένα μήνυμα λάθους:

```
$email = test_input($_POST["email"]);
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
  $emailErr = "Invalid email format";
}
```

PHP - Επικύρωση URL

Ο παρακάτω κώδικας δείχνει ένα τρόπο για να ελέγξετε αν μια σύνταξη διεύθυνσης URL είναι έγκυρη (αυτή η κανονική έκφραση επιτρέπει επίσης παύλες στο URL). Εάν η σύνταξη διεύθυνσης URL δεν είναι έγκυρη, τότε να αποθηκεύσετε ένα μήνυμα λάθους:

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

PHP - Επικύρωση ονόματος, E-mail και URL

Τώρα, το script μοιάζει με αυτό:

Παράδειγμα (php_59.php)

```
<!DOCTYPE HTML>
<html>
```

```
<head>
<style>
.error {color: #FF0000;}
</style>
</head>
<body>

<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
   if (empty($_POST["name"])) {
     $nameErr = "Name is required";
   } else {
     $name = test_input($_POST["name"]);
     // check if name only contains letters and whitespace
     if (!preg_match("/^[a-zA-Z ]*$/",$name)) {
       $nameErr = "Only letters and white space allowed";      }
   }

   if (empty($_POST["email"])) {
     $emailErr = "Email is required";
   } else {
     $email = test_input($_POST["email"]);
     // check if e-mail address is well-formed
     if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
       $emailErr = "Invalid email format";      }
   }

   if (empty($_POST["website"])) {
     $website = "";
   } else {
     $website = test_input($_POST["website"]);
     // check if URL address syntax is valid (this regular expression also allows dashes
in the URL)
     if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-
9+&@#\/%?=~_|!:,.;]*[-a-z0-9+&@#\/%=~_|]/i",$website)) {
       $websiteErr = "Invalid URL";      }   }

   if (empty($_POST["comment"])) {
     $comment = "";
   } else {
     $comment = test_input($_POST["comment"]);    }

   if (empty($_POST["gender"])) {
     $genderErr = "Gender is required";
   } else {
     $gender = test_input($_POST["gender"]);   }
}

function test_input($data) {
```

```
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>

<h2>PHP Form Validation Example</h2>
<p><span class="error">* required field.</span></p>
<form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">
  Name: <input type="text" name="name">
  <span class="error">* <?php echo $nameErr;?></span>
  <br><br>
  E-mail: <input type="text" name="email">
  <span class="error">* <?php echo $emailErr;?></span>
  <br><br>
  Website: <input type="text" name="website">
  <span class="error"><?php echo $websiteErr;?></span>
  <br><br>
  Comment: <textarea name="comment" rows="5" cols="40"></textarea>
  <br><br>
  Gender:
  <input type="radio" name="gender" value="female">Female
  <input type="radio" name="gender" value="male">Male
  <span class="error">* <?php echo $genderErr;?></span>
  <br><br>
  <input type="submit" name="submit" value="Submit">
</form>

<?php
echo "<h2>Your Input:</h2>";
echo $name;
echo "<br>";
echo $email;
echo "<br>";
echo $website;
echo "<br>";
echo $comment;
echo "<br>";
echo $gender;
?>

</body>
</html>
```

Το επόμενο βήμα είναι να δείξουμε πώς να αποτρέψουμε τη φόρμα από το άδειασμα όλων των πεδίων εισαγωγής, όταν ο χρήστης υποβάλλει τη φόρμα.

PHP Παράδειγμα συμπλήρωσης φόρμας

Αυτό το κεφάλαιο δείχνει πώς να κρατήσετε την εμφάνιση των τιμών στα πεδία εισαγωγής, όταν ο χρήστης πατήσει το κουμπί υποβολής.

PHP - Κρατήστε τις τιμές στη φόρμα

Για να εμφανίσουμε τις τιμές στα πεδία εισαγωγής αφού ο χρήστης πατήσει το κουμπί υποβολής, προσθέτουμε ένα μικρό PHP script στην τιμή των ακόλουθων πεδίων εισόδου: όνομα, e-mail, και την ιστοσελίδα. Στο πεδίο σχόλιο textarea, βάζουμε το script μεταξύ των <textarea> και </ textarea> tags. Το μικρό script αποδίδει την τιμή των μεταβλητών $ name, $ e-mail, $website, και $comment.

Στη συνέχεια, θα πρέπει επίσης να δείξουμε ποιο κουμπί ελέγχθηκε. Για το σκοπό αυτό, θα πρέπει να χειριστούμε το χαρακτηριστικό checked (όχι το χαρακτηριστικό value για τα κουμπιά επιλογής):

```
Name: <input type="text" name="name" value="<?php echo $name;?>">

E-mail: <input type="text" name="email" value="<?php echo $email;?>">

Website: <input type="text" name="website" value="<?php echo $website;?>">

Comment: <textarea name="comment" rows="5" cols="40"><?php echo
$comment;?></textarea>

Gender:
<input type="radio" name="gender"
<?php if (isset($gender) && $gender=="female") echo "checked";?>
value="female">Female
<input type="radio" name="gender"
<?php if (isset($gender) && $gender=="male") echo "checked";?>
value="male">Male
```

Εδώ είναι ο πλήρης κώδικας για το παράδειγμα επικύρωσης PHP φόρμας:

Παράδειγμα (php_60.php)

```
<!DOCTYPE HTML>
<html>
```

```
<head>
<style>
.error {color: #FF0000;}
</style>
</head>
<body>

<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
   if (empty($_POST["name"])) {
     $nameErr = "Name is required";
   } else {
     $name = test_input($_POST["name"]);
     // check if name only contains letters and whitespace
     if (!preg_match("/^[a-zA-Z ]*$/",$name)) {
       $nameErr = "Only letters and white space allowed";
     }   }

   if (empty($_POST["email"])) {
     $emailErr = "Email is required";
   } else {
     $email = test_input($_POST["email"]);
     // check if e-mail address is well-formed
     if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
       $emailErr = "Invalid email format";
     }   }

   if (empty($_POST["website"])) {
     $website = "";
   } else {
     $website = test_input($_POST["website"]);
     // check if URL address syntax is valid (this regular expression also allows dashes
in the URL)
     if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-
9+&@#\/%?=~_|!:,.;]*[-a-z0-9+&@#\/%=~_|]/i",$website)) {
       $websiteErr = "Invalid URL";
     }   }

   if (empty($_POST["comment"])) {
     $comment = "";
   } else {
     $comment = test_input($_POST["comment"]);
   }

   if (empty($_POST["gender"])) {
     $genderErr = "Gender is required";
   } else {
     $gender = test_input($_POST["gender"]);   }
}
```

```
function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>

<h2>PHP Form Validation Example</h2>
<p><span class="error">* required field.</span></p>
<form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">
  Name: <input type="text" name="name" value="<?php echo $name;?>">
  <span class="error">* <?php echo $nameErr;?></span>
  <br><br>
  E-mail: <input type="text" name="email" value="<?php echo $email;?>">
  <span class="error">* <?php echo $emailErr;?></span>
  <br><br>
  Website: <input type="text" name="website" value="<?php echo $website;?>">
  <span class="error"><?php echo $websiteErr;?></span>
  <br><br>
  Comment: <textarea name="comment" rows="5" cols="40"><?php echo
$comment;?></textarea>
  <br><br>
  Gender:
  <input type="radio" name="gender" <?php if (isset($gender) &&
$gender=="female") echo "checked";?>  value="female">Female
  <input type="radio" name="gender" <?php if (isset($gender) &&
$gender=="male") echo "checked";?>  value="male">Male
  <span class="error">* <?php echo $genderErr;?></span>
  <br><br>
  <input type="submit" name="submit" value="Submit">
</form>

<?php
echo "<h2>Your Input:</h2>";
echo $name;
echo "<br>";
echo $email;
echo "<br>";
echo $website;
echo "<br>";
echo $comment;
echo "<br>";
echo $gender;
?>

</body>
</html>
```

PHP πολυδιάστατοι πίνακες

Νωρίτερα, έχουμε περιγράψει πίνακες που είναι μια λίστα ζευγών κλειδιού / τιμής.

Ωστόσο, μερικές φορές θέλετε να αποθηκεύσετε τιμές με περισσότερα από ένα κλειδιά.

Αυτό μπορεί να γίνει με πολυδιάστατους πίνακες.

PHP - πολυδιάστατοι πίνακες

Ένας πολυδιάστατος πίνακα είναι ένας πίνακας που περιέχει έναν ή περισσότερους πίνακες.

Η PHP καταλαβαίνει ότι οι πολυδιάστατοι πίνακες είναι δύο, τρία, τέσσερα, πέντε, ή περισσότερα επίπεδα βάθος. Ωστόσο, πίνακες με περισσότερα από τρία επίπεδα εις βάθος είναι δύσκολο για τους περισσότερους ανθρώπους να τους διαχειριστούν,

> **Η διάσταση του πίνακα δείχνει τον αριθμό των δεικτών που χρειάζεστε για να επιλέξετε ένα στοιχείο.**

- Για ένα δισδιάστατο πίνακα θα πρέπει να έχετε δύο δείκτες για να επιλέξετε ένα στοιχείο
- Για ένα τρισδιάστατο πίνακα θα πρέπει να έχετε τρεις δείκτες για να επιλέξετε ένα στοιχείο

PHP – δισδιάστατοι πίνακες

Ένας δισδιάστατος πίνακας είναι ένας πίνακας από πίνακες (ένας τρισδιάστατος πίνακας είναι ένας πίνακας από πίνακες πινάκων).

Δείτε το παρακάτω πίνακα:

Name	Stock	Sold
Volvo	22	18
BMW	15	13
Saab	5	2
Land Rover	17	15

Μπορούμε να αποθηκεύσουμε τα δεδομένα από τον παραπάνω πίνακα σε ένα δισδιάστατο πίνακα, όπως αυτόν:

```
$cars = array
  (
  array("Volvo",22,18),
  array("BMW",15,13),
  array("Saab",5,2),
  array("Land Rover",17,15)
  );
```

Τώρα ο δισδιάστατος πίνακας $cars περιέχει τέσσερις πίνακες, και έχει δύο δείκτες: γραμμή και στήλη.

Για να αποκτήσετε πρόσβαση στα στοιχεία του $cars πρέπει να δείξουμε με τους δύο δείκτες (γραμμή και στήλη).

Παράδειγμα (php_61.php)

```
<!DOCTYPE html>
<html>
<body>

<?php
$cars = array
  (
  array("Volvo",22,18),
  array("BMW",15,13),
  array("Saab",5,2),
  array("Land Rover",17,15)
  );

echo $cars[0][0].": In stock: ".$cars[0][1].", sold: ".$cars[0][2].".<br>";
echo $cars[1][0].": In stock: ".$cars[1][1].", sold: ".$cars[1][2].".<br>";
echo $cars[2][0].": In stock: ".$cars[2][1].", sold: ".$cars[2][2].".<br>";
echo $cars[3][0].": In stock: ".$cars[3][1].", sold: ".$cars[3][2].".<br>";
?>

</body>
</html>
```

Έξοδος:
Volvo: In stock: 22, sold: 18.
BMW: In stock: 15, sold: 13.
Saab: In stock: 5, sold: 2.
Land Rover: In stock: 17, sold: 15.

Μπορούμε επίσης να βάλουμε ένα for βρόχο μέσα σε έναν άλλο for βρόχο για να πάρει τα στοιχεία του πίνακα $cars (πάλι πρέπει να δώσουμε τιμές στους δύο δείκτες):

Παράδειγμα (php_62.php)

```
<!DOCTYPE html>
<html>
<body>

<?php
$cars = array
  (
  array("Volvo",22,18),
  array("BMW",15,13),
  array("Saab",5,2),
  array("Land Rover",17,15)
  );

for ($row = 0; $row <  4; $row++) {
  echo "<p><b>Row number $row</b></p>";
  echo "<ul>";
  for ($col = 0; $col <  3; $col++) {
    echo "<li>".$cars[$row][$col]."</li>";
  }
  echo "</ul>";
}
?>

</body>
</html>
```

Έξοδος:

Row number 0
- Volvo
- 22
- 18

Row number 1
- BMW
- 15
- 13

Row number 2
- Saab
- 5
- 2

Row number 3
- Land Rover
- 17
- 15

PHP Cookies

Ένα cookie χρησιμοποιείται συχνά για να προσδιορίσει/αναγνωρίσει ένα χρήστη.

Τι είναι το Cookie;

Ένα cookie είναι ένα μικρό αρχείο που ο εξυπηρετητής (server) ενσωματώνει στον υπολογιστή του χρήστη. Κάθε φορά που ο ίδιος υπολογιστής ζητά μια σελίδα με ένα πρόγραμμα περιήγησης, θα στείλει το cookie επίσης. Με την PHP, μπορείτε να δημιουργήσετε αλλά και να ανακτήσετε τις τιμές των cookies.

Δημιουργήστε cookies με την PHP

Ένα cookie (μπισκότο) δημιουργείται με τη συνάρτηση setcookie().

Σύνταξη

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

Μόνο η παράμετρος *όνομα* απαιτείται. Όλες οι άλλες παράμετροι είναι προαιρετικές.

PHP Δημιουργία / Ανάκτηση ενός cookie

Το ακόλουθο παράδειγμα δημιουργεί ένα cookie που ονομάζεται "user" με την τιμή "John Doe". Το cookie θα λήξει μετά από 30 ημέρες (86400 * 30). Το "/" σημαίνει ότι το cookie είναι διαθέσιμο σε ολόκληρο το δικτυακό τόπο (αλλιώς, επιλέξτε το φάκελο που προτιμάτε).

Στη συνέχεια ανακτούμε την τιμή του cookie "user" (χρησιμοποιώντας την καθολική μεταβλητή $ _COOKIE). Μπορούμε επίσης να χρησιμοποιήσουμε τη λειτουργία isset() για να μάθουμε αν το cookie έχει οριστεί:

Παράδειγμα (php_63.php)

```
<!DOCTYPE html>
<?php
$cookie_name = "user";
$cookie_value = "John Doe";
```

```
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); // 86400 = 1
day
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

<p><strong>Note:</strong> You might have to reload the page to see the value of
the cookie.</p>

</body>
</html>
```

Έξοδος:

Cookie 'user' is set!
Value is: John Doe

Note: You might have to reload the page to see the value of the cookie.

 **Σημείωση:** Η συνάρτηση setcookie() πρέπει να εμφανίζεται πριν το <html> tag.

Σημείωση: Η τιμή του cookie αυτόματα κωδικοποιείται σε url όταν αποστέλλεται το cookie, και αυτόματα αποκωδικοποιείται κατά την παραλαβή του.

Τροποποίηση τιμής ενός cookie

Για να τροποποιήσετε ένα cookie, απλά ορίστε (και πάλι) το cookie χρησιμοποιώντας τη συνάρτηση setcookie():

Παράδειγμα (php_64.php)

```
<!DOCTYPE html>
<?php
$cookie_name = "user";
$cookie_value = "Alex Porter";
```

```
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

<p><strong>Note:</strong> You might have to reload the page to see the new value of the cookie.</p>

</body>
</html>
```

Διαγραφή Cookie

Για να διαγράψετε ένα cookie, χρησιμοποιήστε τη συνάρτηση setcookie() με ημερομηνία λήξης στο παρελθόν:

Παράδειγμα (php_65.php)

```
<!DOCTYPE html>
<?php
// set the expiration date to one hour ago
setcookie("user", "", time() - 3600);
?>
<html>
<body>

<?php
echo "Cookie 'user' is deleted.";
?>

</body>
</html>
```

Ελέγξτε εάν είναι ενεργοποιημένα τα cookies

Το ακόλουθο παράδειγμα δημιουργεί ένα μικρό σενάριο που ελέγχει αν τα cookies είναι ενεργοποιημένα. Πρώτα, προσπαθήστε να δημιουργήσετε ένα δοκιμαστικό cookie με τη συνάρτηση setcookie(), Μετά μετρήστε τη μεταβλητή τύπου πίνακα $ _COOKIE:

Παράδειγμα (php_66.php)

```
<!DOCTYPE html>
<?php
setcookie("test_cookie", "test", time() + 3600, '/');
?>
<html>
<body>

<?php
if(count($_COOKIE) > 0) {
    echo "Cookies are enabled.";
} else {
    echo "Cookies are disabled.";
}
?>

</body>
</html>
```

PHP Συνεδρίες (sessions)

Μια συνεδρία είναι ένας τρόπος για να αποθηκεύσετε τις πληροφορίες (σε μεταβλητές) που θα χρησιμοποιηθούν σε πολλαπλές σελίδες.

Σε αντίθεση με ένα cookie, οι πληροφορίες δεν αποθηκεύονται στον υπολογιστή των χρηστών.

Τι είναι μια PHP Σύνοδος;

Όταν εργάζεστε με μια εφαρμογή, μπορείτε να την ανοίξετε, να κάνετε κάποιες αλλαγές, και στη συνέχεια να την κλείσετε. Αυτό είναι σαν μια συνεδρία. Ο υπολογιστής δεν ξέρει ποιος είστε. Ξέρει πότε θα ξεκινήσει η εφαρμογή και πότε θα τελειώσει. Αλλά στο διαδίκτυο υπάρχει ένα πρόβλημα: ο web server δεν ξέρει ποιος είστε ή τι κάνετε, επειδή η διεύθυνση HTTP δεν διατηρεί την κατάσταση.

Οι μεταβλητές συνεδρίας λύνουν αυτό το πρόβλημα με την αποθήκευση των πληροφοριών των χρηστών που θα χρησιμοποιηθούν σε πολλές σελίδες (π.χ. όνομα χρήστη, το αγαπημένο χρώμα, κλπ.). Από προεπιλογή, οι μεταβλητές συνόδου διαρκούν μέχρι ο χρήστης να κλείσει το πρόγραμμα περιήγησης.

Επομένως οι μεταβλητές συνεδρίας κατέχουν πληροφορίες για ένα μόνο χρήστη, και είναι διαθέσιμες σε όλες τις σελίδες σε μια εφαρμογή.

Συμβουλή: Εάν χρειάζεστε μια μόνιμη αποθήκευση, μπορεί να θέλετε να αποθηκεύσετε τα δεδομένα σε μια βάση δεδομένων .

Ξεκινήστε μια PHP συνεδρία

Μια συνεδρία ξεκινά με τη συνάρτηση session_start().

Οι μεταβλητές συνόδου ορίζονται με την καθολική μεταβλητή της PHP, $ _SESSION.

Τώρα, ας δημιουργήσουμε μια νέα σελίδα Σε αυτή τη σελίδα, ξεκινάμε μια νέα PHP σύνοδο και θέτουμε κάποιες μεταβλητές συνόδου:

Παράδειγμα (php_67.php)

```
<?php
// Start the session
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>

</body>
</html>
```

Σημείωση: Η συνάρτηση session_start() πρέπει να είναι το πρώτο πράγμα στο έγγραφό σας. Πριν από τυχόν ετικέτες HTML.

Πάρτε τιμές μεταβλητών συνεδρίας PHP

Στη συνέχεια, έχουμε δημιουργήσει μια άλλη σελίδα. Από τη σελίδα αυτή, θα αποκτήσουμε πρόσβαση στις πληροφορίες συνεδρίας που θέσαμε στην προηγούμενη σελίδα.

Παρατηρήστε ότι οι μεταβλητές συνεδρίας δεν έχουν περάσει μεμονωμένα σε κάθε νέα σελίδα, αλλά ανακτώνται από τη σύνοδο που ανοίγουμε στην αρχή της κάθε σελίδας (session_start()).

Παρατηρήστε, επίσης, ότι όλες οι τιμές μεταβλητών συνόδου είναι αποθηκευμένες στην καθολική $ _SESSION μεταβλητή:

Παράδειγμα (php_68.php)

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Echo session variables that were set on previous page
echo "Favorite color is " . $_SESSION["favcolor"] . ".<br>";
echo "Favorite animal is " . $_SESSION["favanimal"] . ".";
?>

</body>
</html>
```

Ένας άλλος τρόπος για να δείξουμε όλες τις τιμές μεταβλητών συνόδου για μια συνεδρία χρήστη είναι να εκτελέσουμε τον ακόλουθο κώδικα:

Παράδειγμα (php_69.php)

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

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

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

Πώς λειτουργεί; Πώς ξέρω ότι είμαι εγώ;

Οι περισσότερες συνεδρίες ορίζουν ένα κλειδί-χρήστη στον υπολογιστή του χρήστη που μοιάζει κάτι σαν αυτό: 765487cf34ert8dede5a562e4f3a7e12. Στη συνέχεια, όταν μια συνεδρία ανοίγει σε μια άλλη σελίδα, σαρώνει (ο server) τον υπολογιστή για ένα κλειδί-χρήστη. Αν υπάρχει ταίριασμα, αποκτά πρόσβαση στη συνεδρία, αν όχι, ξεκινά μια νέα συνεδρία.

Τροποποίηση PHP μεταβλητών συνεδρίας

Για να αλλάξετε μια μεταβλητή συνόδου, απλά γράψτε πάνω της:

Παράδειγμα (php_70.php)

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// to change a session variable, just overwrite it
$_SESSION["favcolor"] = "yellow";
print_r($_SESSION);
?>

</body>
</html>
```

Διακόψτε μια PHP συνεδρία

Για να αφαιρέσετε όλες τις καθολικές μεταβλητές συνεδρίας και να διακόψετε τη συνεδρία, χρησιμοποιήστε τη session_unset() και τη session_destroy():

Παράδειγμα (php_71.php)

```
<?php
session_start();
?>
<!DOCTYPE html>
```

```
<html>
<body>

<?php
// remove all session variables
session_unset();

// destroy the session
session_destroy();

echo "All session variables are now removed, and the session is destroyed."
?>

</body>
</html>
```

PHP MySQL βάση δεδομένων

Με την PHP, μπορείτε να συνδεθείτε και να χειριστείτε τις βάσεις δεδομένων.

Η MySQL είναι το πιο δημοφιλές σύστημα βάσης δεδομένων που χρησιμοποιείται με την PHP.

Τι είναι η MySQL;

- είναι ένα σύστημα βάσης δεδομένων που χρησιμοποιείται στο διαδίκτυο
- είναι ένα σύστημα βάσης δεδομένων που τρέχει σε ένα διακομιστή
- είναι ιδανική για μικρές και μεγάλες εφαρμογές
- είναι πολύ γρήγορη, αξιόπιστη και εύκολη στη χρήση
- χρησιμοποιεί το πρότυπο SQL
- υποστηρίζεται από μια σειρά από πλατφόρμες/λειτουργικά συστήματα
- είναι ελεύθερη να τη κατεβάσετε και να τη χρησιμοποιήσετε
- αναπτύσσεται, διανέμεται και υποστηρίζεται από την Oracle Corporation
- ονομάστηκε από την κόρη (My) του συν-ιδρυτή Monty Widenius

Τα δεδομένα σε μια βάση δεδομένων MySQL αποθηκεύονται σε πίνακες. Ένας πίνακας είναι μια συλλογή των σχετικών δεδομένων, και αποτελείται από στήλες και γραμμές.

Οι βάσεις δεδομένων είναι χρήσιμες για την αποθήκευση πληροφοριών σε κατηγορίες. Μια εταιρεία μπορεί να έχει μια βάση δεδομένων με τους ακόλουθους πίνακες:

- Εργαζόμενοι
- Προϊόντα
- Πελάτες
- Παραγγελίες

PHP + MySQL βάση δεδομένων του συστήματος

- Η PHP σε συνδυασμό με τη MySQL είναι cross-platform λογισμικό (μπορείτε να αναπτύξετε κώδικα στα Windows και να εξυπηρετείται αυτός από έναν server σε μια πλατφόρμα Unix)

Ερωτήματα στη Βάση Δεδομένων

Ένα ερώτημα είναι ένα αίτημα. Μπορούμε να αναζητήσουμε μια βάση δεδομένων για συγκεκριμένες πληροφορίες και να έχουμε ένα σύνολο επιστρεφόμενων εγγραφών.

Δείτε το παρακάτω ερώτημα (χρησιμοποιώντας τυπικές SQL):

```
SELECT LastName FROM Employees;
```

Η παραπάνω ερώτηση επιλέγει όλα τα στοιχεία στη στήλη "LastName" από τον πίνακα "Employees". Περισσότερα σχετικά με τη SQL (φροντιστήριο SQL).

Κατεβάστε τη βάση δεδομένων MySQL

Αν δεν έχετε ένα διακομιστή PHP με μια βάση δεδομένων MySQL, μπορείτε να τα κατεβάσετε δωρεάν εδώ: http://www.mysql.com

---

Σχετικά με τη βάση δεδομένων MySQL

Η MySQL είναι το πρότυπο σύστημα βάσης δεδομένων de-facto για ιστοσελίδες, με τεράστιους όγκους δεδομένων και τελικών χρηστών (όπως το Facebook, το Twitter και τη Wikipedia).

Ένα άλλο σημαντικό πράγμα για τη MySQL είναι ότι μπορεί να υποστεί ‘μείωση κλίμακας’ (scaled down) ώστε να υποστηρίζει τις ενσωματωμένες (embedded) εφαρμογές βάσεων δεδομένων (πχ. σε mobile περιβάλλον).

Κοιτάξτε στη διεύθυνση http://www.mysql.com/customers/ για μια επισκόπηση των επιχειρήσεων που χρησιμοποιούν MySQL.

PHP Σύνδεση σε MySQL

Η PHP μπορεί να λειτουργήσει με μια βάση δεδομένων MySQL χρησιμοποιώντας:

- **MySQLi επέκταση** (το "i" σημαίνει βελτιωμένη)
- **PDO (PHP Data Objects)**

Οι παλαιότερες εκδόσεις της PHP χρησιμοποίησαν την (απλή) επέκταση (extension) MySQL. Ωστόσο, αυτή καταργήθηκε το 2012.

Πρέπει να χρησιμοποιήσω mysqli ή PDO;

Εάν χρειάζεστε μια σύντομη απάντηση, θα ήταν "Ότι θέλετε".

Τόσο η mysqli όσο και η PDO έχουν τα πλεονεκτήματά τους:

Η ‘σύνταξη εντολών’ PDO θα λειτουργήσει σε 12 διαφορετικά συστήματα βάσεων δεδομένων, ενώ η mysqli θα λειτουργήσει μόνο με βάσεις δεδομένων MySQL.

Έτσι, αν έχετε να αλλάξετε το πρόγραμμά σας για να χρησιμοποιήσετε μια άλλη βάση δεδομένων, η PDO καθιστά τη διαδικασία πιο εύκολη. Το μόνο που χρειάζεται να αλλάξετε είναι η συμβολοσειρά σύνδεσης και μερικά ερωτήματα. Με τη mysqli, θα χρειαστεί να ξαναγράψετε το σύνολο των ερωτημάτων-κώδικα που περιλαμβάνονται.

Και οι δύο είναι αντικειμενοστρεφείς, αλλά η mysqli προσφέρει επίσης μια διαδικαστική προσέγγιση (procedural API).

Και οι δύο υποστηρίζουν “Έτοιμες Δηλώσεις”. Οι “Έτοιμες Δηλώσεις” προστατεύουν από SQL injection, και είναι πολύ σημαντικές για την ασφάλεια των εφαρμογών web.

MySQL Παραδείγματα (και με σύνταξη mysqli και με σύνταξη PDO)

Σε αυτό, και στα επόμενα κεφάλαια θα παρουσιάσουμε τρεις τρόπους εργασίας με PHP και MySQL:

- Mysqli object-oriented (αντικειμενοστρεφής)
- Mysqli (διαδικαστική)
- PDO

Mysqli Εγκατάσταση

Για Linux και Windows: Η mysqli επέκταση εγκαθίσταται αυτόματα στις περισσότερες περιπτώσεις, όταν εγκατασταθεί το πακέτο PHP MySQL.

Για λεπτομέρειες σχετικά με την εγκατάσταση, πηγαίνετε στο:
http://php.net/manual/en/mysqli.installation.php

PDO Εγκατάσταση

Η PDO υποστήριξη εγκαθίσταται αυτόματα στις περισσότερες περιπτώσεις, όταν εγκατασταθεί το πακέτο PHP MySQL.Για λεπτομέρειες σχετικά με την εγκατάσταση, πηγαίνετε στο: http://php.net/manual/en/pdo.installation.php

---

Στα παρακάτω παραδείγματα υποθέτουμε ότι υπάρχει στη MySQL χρήστης **root** με password κενό, ο οποίος είναι ο διαχειριστής της.

.

Ανοίξτε μια σύνδεση με MySQL

Πριν μπορέσουμε να έχουμε πρόσβαση στα δεδομένα σε μια βάση δεδομένων τύπου MySQL, πρέπει να είμαστε σε θέση να συνδεθούμε με το διακομιστή:

Παράδειγμα PHP_SQL1.php (mysqli αντικειμενοστραφής)

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

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

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

Σημείωση σχετικά με το αντικειμενοστρεφές παραπάνω παράδειγμα: η $connect_error είχε σπάσει μέχρι την PHP 5.2.9 και 5.3.0. Εάν πρέπει να διασφαλιστεί η συμβατότητα με τις εκδόσεις της PHP πριν την 5.2.9 και 5.3.0, χρησιμοποιήστε τον ακόλουθο κώδικα:

```
// Check connection
If (mysqli_connect_error ()) {
    die ("Database connection failed:". mysqli_connect_error ()); }
```

Με σωστές τιμές στις μεταβλητές $username και $password, θα εμφανιστεί μήνυμα επιτυχούς σύνδεσης. Αν κάποια τιμή δεν είναι σωστή (πχ. $user=root και $password=root, στο PHP_SQL1b.php), τότε θα εμφανιστεί μήνυμα ΑΠΟΤΥΧΙΑΣ για τη σύνδεση:

Connection failed: Access denied for user 'root'@'localhost' (using password: YES)

Τα παρακάτω παραδείγματα είναι τα αντίστοιχα του 1, με διαδικαστική mysqli και με χρήση PDO αντίστοιχα. Με κόκκινους χαρακτήρες είναι η διαφορετική σύνταξη.

Παράδειγμα PHP_SQL2.php (mysqli διαδικαστική)

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

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

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

Και εδώ οι λάθος τιμές σε $username και $password (πχ. PHP_SQL2b.php) προκαλούν εμφάνιση μηνύματος ΑΠΟΤΥΧΙΑΣ σύνδεσης.

Παράδειγμα PHP_SQL3.php (PDO)

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

try {
   $conn = new PDO("mysql:host=$servername;dbname=test", $username, $password);
   // set the PDO error mode to exception
```

```
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    echo "Connected successfully";
    }
catch(PDOException $e)
    {
    echo "Connection failed: " . $e->getMessage();
    }
?>
```

Παρατηρήστε ότι στο παραπάνω παράδειγμα PDO έχουμε επίσης καθορίσει μια βάση δεδομένων (test). Υποθέτουμε ότι υπάρχει ήδη στη MySQL μια τέτοια βάση ήδη κατασκευασμένη που επιτρέπει στον user να συνδεθεί. Αν όχι, τότε με το εργαλείο phpMyAdmin δημιουργούμε μια βάση test και θέτουμε τη μεταβλητή $username=root για να δοκιμάσουμε το PHP_SQL3.php.
Η PDO απαιτεί μια έγκυρη βάση δεδομένων για να συνδεθείτε. Εάν δεν υπάρχει βάση δεδομένων έχει καθοριστεί, μια εξαίρεση. Δείτε πχ. το PHP_SQL3b.php όπου ως βάση δίνεται η ανύπαρκτη 'dontknow'.

Συμβουλή: Ένα μεγάλο πλεονέκτημα της PDO είναι ότι έχει μια κλάση εξαίρεσης για να χειριστεί τυχόν προβλήματα που μπορεί να εμφανιστούν σε ερωτήματα της βάσης δεδομένων μας. Εάν μια εξαίρεση παγιδευτεί εντός του try {} μπλοκ, το σενάριο σταματά την εκτέλεση και ρέει κατευθείαν στο πρώτο catch () {} μπλοκ.

Κλείστε τη σύνδεση

Η σύνδεση αυτή θα κλείνει αυτόματα όταν τελειώνει το σενάριο. Για να κλείσετε τη σύνδεση πριν, χρησιμοποιήστε τα ακόλουθα:

Παράδειγμα (mysqli αντικειμενοστρεφής)

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

Παράδειγμα (mysqli διαδικαστική)

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

Παράδειγμα (PDO)

```
$conn = null;
```

PHP Δημιουργήστε μια βάση δεδομένων MySQL

Μια βάση δεδομένων αποτελείται από έναν ή περισσότερους πίνακες.

Θα χρειαστείτε ειδικά δικαιώματα CREATE για να δημιουργήσετε ή να διαγράψετε μια βάση δεδομένων MySQL. Εξ' ορισμού, τέτοια δικαιώματα έχει ο διαχειριστής root της MySQL.

Δημιουργήστε μια βάση δεδομένων MySQL χρησιμοποιώντας mysqli και PDO

Η CREATE DATABASE δήλωση χρησιμοποιείται για να δημιουργήσει μια βάση δεδομένων σε MySQL.

Τα παρακάτω παραδείγματα δημιουργούν μια βάση δεδομένων που ονομάζεται "myDB":

Παράδειγμα PHP_SQL4.php (mysqli αντικειμενοστραφής)

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

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

// Create database
$sql = "CREATE DATABASE myDB";
if ($conn->query($sql) === TRUE) {
   echo "Database created successfully";
} else {
   echo "Error creating database: " . $conn->error;
}

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

Σημείωση: Όταν δημιουργείτε μια νέα βάση δεδομένων, πρέπει να καθορίσετε μόνο τις τρεις πρώτες παραμέτρους για το mysqli αντικείμενο (ServerName, username, password).

Συμβουλή: Εάν πρέπει να χρησιμοποιήσετε μια συγκεκριμένη θύρα, προσθέστε μια κενή συμβολοσειρά για την παράμετρο του ονόματος της βάσης δεδομένων, όπως αυτό: *new mysqli ("localhost", "username", "password", "", port);*

Το μήνυμα «*Database created successfully*» θα εμφανισθεί αν γίνει επιτυχώς η δημιουργία της βάσης myDB. Αν ήδη υπάρχει η βάση myDB, φυσικά δεν μπορεί να ξανα—ημιουργηθεί, και θα εμφανιστεί το ακόλουθο:

Error creating database: Can't create database 'mydb'; database exists

Αν δεν έχει δικαιώματα ο χρήστης να δημιουργήσει μια βάση δεδομένων (δείτε το PHP_SQL4b.php, όπου $username=user), τότε θα εμφανισθεί μήνυμα όπως:

Error creating database: Access denied for user ''@'localhost' to database 'mydb'

Παράδειγμα PHP_SQL5.php (mysqli διαδικαστική)

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

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

// Create database
$sql = "CREATE DATABASE myDB";
if (mysqli_query($conn, $sql)) {
   echo "Database created successfully";
} else {
   echo "Error creating database: " . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

Για να δοκιμάσετε και τη δυνατότητα δημιουργίας βάσης από το προηγούμενο παράδειγμα ή βάλτε άλλο όνομα βάσης, πχ. newMyDB, ή σβήστε τη βάση mydb (πχ. από το phpMyAdmin επιλέξτε τη βάση mydb και από τη καρτέλα Λειτουργίες επιλέξτε ‘Διαγραφή της βάσης δεδομένων (DROP)’ που εκτελεί την DROP DATABASE mydb;)

Σημείωση: Το παρακάτω παράδειγμα δημιουργεί μια βάση δεδομένων που ονομάζεται "myDBPDO":

Παράδειγμα PHP_SQL6.php (PDO)

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

try {
   $conn = new PDO("mysql:host=$servername;dbname=myDB", $username,
$password);
   // set the PDO error mode to exception
   $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
   $sql = "CREATE DATABASE myDBPDO";
   // use exec() because no results are returned
   $conn->exec($sql);
   echo "Database created successfully<br>";
   }
catch(PDOException $e)
   {
   echo $sql . "<br>" . $e->getMessage();
   }

$conn = null;
?>
```

Συμβουλή: Στο παραπάνω catch μπλοκ, εμφανίζουμε την πρόταση SQL και το παραγόμενο μήνυμα σφάλματος.

PHP Δημιουργία πινάκων MySQL

Ένας πίνακας βάσης δεδομένων έχει το δικό του μοναδικό όνομα και αποτελείται από στήλες και γραμμές.

Δημιουργήστε ένα πίνακα MySQL χρησιμοποιώντας mysqli και PDO

Η δήλωση CREATE TABLE χρησιμοποιείται για να δημιουργήσετε έναν πίνακα σε MySQL.

Θα δημιουργήσουμε έναν πίνακα που ονομάζεται «MyGuests», με πέντε στήλες: "id", "firstname", "lastname", "e-mail" και "reg_date":

```
CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP
)
```

Σημειώσεις για τον παραπάνω πίνακα:

Ο τύπος δεδομένων καθορίζει το είδος των δεδομένων που η στήλη μπορεί να κρατήσει. Για μια πλήρη αναφορά όλων των διαθέσιμων τύπων δεδομένων, μεταβείτε στη διεύθυνση αναφορά SQL τύπων δεδομένων.

Μετά από τον τύπο των δεδομένων, μπορείτε να καθορίσετε άλλα προαιρετικά χαρακτηριστικά για κάθε στήλη:

- NOT NULL - Κάθε γραμμή πρέπει να περιέχει μια τιμή για τη στήλη αυτή, οι τιμές null δεν επιτρέπονται
- Προεπιλεγμένη τιμή - Ορίστε μια προεπιλεγμένη τιμή που προστίθεται όταν καμία άλλη τιμή δεν έχει περάσει
- UNSIGNED - Χρησιμοποιείται για τους τύπους αριθμών: περιορίζει τα αποθηκευμένα δεδομένα για θετικούς αριθμούς και το μηδέν
- AUTO INCREMENT – Η MySQL αυξάνει αυτόματα την τιμή του πεδίου κατά 1 κάθε φορά που μια νέα εγγραφή προστίθεται
- PRIMARY KEY - Χρησιμοποιείται για να ορίσει τη στήλη (πεδίο) ως πρωτεύον κλειδί, δηλαδή μια στήλη που προσδιορίζει σε ποια γραμμή του πίνακα αναφερόμαστε. Η στήλη με τη ρύθμιση PRIMARY KEY είναι συχνά ένας αριθμός ID, και χρησιμοποιείται συχνά με AUTO_INCREMENT

Κάθε πίνακας πρέπει να έχει ένα πρωτεύον κλειδί στήλη (στην προκειμένη περίπτωση: η στήλη "id"). Η τιμή του πρέπει να είναι μοναδική για κάθε γραμμή (εγγραφή) στον πίνακα.

Τα παρακάτω παραδείγματα δείχνουν πώς να δημιουργήσετε πίνακα στην PHP:

Παράδειγμα PHP_SQL7.php (mysqli αντικειμενοστραφής)

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

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

// sql to create table
$sql = "CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP
)";

if ($conn->query($sql) === TRUE) {
   echo "Table MyGuests created successfully";
} else {
   echo "Error creating table: " . $conn->error;
}

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

Παράδειγμα PHP_SQL8.php (mysqli διαδικαστική)

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

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
```

```
    die("Connection failed: " . mysqli_connect_error());
}

// sql to create table
$sql = "CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP
)";

if (mysqli_query($conn, $sql)) {
    echo "Table MyGuests created successfully";
} else {
    echo "Error creating table: " . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

Παράδειγμα PHP_SQL9.php (PDO)

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

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

    // sql to create table
    $sql = "CREATE TABLE MyGuests (
    id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    firstname VARCHAR(30) NOT NULL,
    lastname VARCHAR(30) NOT NULL,
    email VARCHAR(50),
    reg_date TIMESTAMP
    )";

    // use exec() because no results are returned
```

```
    $conn->exec($sql);
    echo "Table MyGuests created successfully";
    }
catch(PDOException $e)
    {
    echo $sql . "<br>" . $e->getMessage();
    }

$conn = null;
?>
```

Εισαγωγή δεδομένων σε MySQL

Εισαγωγή δεδομένων σε MySQL χρησιμοποιώντας mysqli και PDO

Μετά από μια βάση δεδομένων και ένα πίνακα που έχουν δημιουργηθεί, μπορούμε να αρχίσουμε να προσθέτουμε στοιχεία σε αυτούς.

Εδώ είναι μερικοί κανόνες σύνταξης που ακολουθούν:

- Το ερώτημα SQL στη PHP πρέπει να είναι εντός εισαγωγικών
- Οι τιμές συμβολοσειράς εντός του SQL ερωτήματος πρέπει να είναι εντός (μονών) εισαγωγικών
- Αριθμητικές τιμές δεν θα πρέπει να είναι εντός εισαγωγικών
- Η λέξη NULL δεν θα πρέπει να είναι εντός εισαγωγικών

Η INSERT INTO δήλωση χρησιμοποιείται για να προσθέσετε νέες εγγραφές σε έναν πίνακα MySQL:

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

Στο προηγούμενο κεφάλαιο δημιουργήσαμε ένα άδειο πίνακα με το όνομα "MyGuests" με πέντε στήλες: "id", "firstname", "lastname", "email" και "reg_date". Τώρα, ας συμπληρωθεί ο πίνακας με τα δεδομένα.

Σημείωση: Αν μια στήλη είναι AUTO_INCREMENT (όπως η στήλη "id") ή TIMESTAMP (όπως η στήλη "reg_date»), δεν είναι ανάγκη να καθοριστεί στο ερώτημα SQL. Η MySQL θα προσθέσει αυτόματα την τιμή.

Τα ακόλουθα παραδείγματα προσθέτουν μια νέα εγγραφή στο πίνακα "MyGuests":

Παράδειγμα PHP_SQL10.php (mysqli αντικειμενοστραφής)

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

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

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";

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

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

Παράδειγμα PHP_SQL11.php (mysqli διαδικαστική)

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

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

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";
```

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

mysqli_close($conn);
?>
```

Παράδειγμα PHP_SQL12.php (PDO)

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

try {
   $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username,
$password);
   // set the PDO error mode to exception
   $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
   $sql = "INSERT INTO MyGuests (firstname, lastname, email)
   VALUES ('John', 'Doe', 'john@example.com')";
   // use exec() because no results are returned
   $conn->exec($sql);
   echo "New record created successfully";
   }
catch(PDOException $e)
   {
   echo $sql . "<br>" . $e->getMessage();
   }

$conn = null;
?>
```

PHP Πάρτε το ID της Τελευταίας Εγγραφής που εισήχθη

Αν κάνουμε μια προσθήκη ή ενημέρωση σχετικά με έναν πίνακα με ένα πεδίο AUTO_INCREMENT, μπορούμε να πάρουμε το ID της τελευταίας εισηχθήσας / ενημερωμένης εγγραφής άμεσα.

Στον πίνακα "MyGuests", η στήλη "ID" είναι ένα πεδίο AUTO_INCREMENT:

```
CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP
)
```

Τα παρακάτω παραδείγματα είναι ίδια με της προηγούμενης παραγράφου, εκτός από το ότι έχουμε προσθέσει μια γραμμή κώδικα για να ανακτήσουμε το ID της τελευταίας εισαχθείσας εγγραφής. Επίσης το εμφανίζουμε το ID αυτό:

Παράδειγμα PHP_SQL13.php (mysqli αντικειμενοστραφής)

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

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

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";

if ($conn->query($sql) === TRUE) {
    $last_id = $conn->insert_id;
    echo "New record created successfully. Last inserted ID is: " . $last_id;
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
```

```
}

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

Παράδειγμα PHP_SQL14.php (mysqli διαδικαστικά)

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

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

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";

if (mysqli_query($conn, $sql)) {
   $last_id = mysqli_insert_id($conn);
   echo "New record created successfully. Last inserted ID is: " . $last_id;
} else {
   echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

Παράδειγμα PHP_SQL15.php (PDO)

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

try {
   $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username,
$password);
```

```
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $sql = "INSERT INTO MyGuests (firstname, lastname, email)
    VALUES ('John', 'Doe', 'john@example.com')";
    // use exec() because no results are returned
    $conn->exec($sql);
    $last_id = $conn->lastInsertId();
    echo "New record created successfully. Last inserted ID is: " . $last_id;
    }
catch(PDOException $e)
    {
    echo $sql . "<br>" . $e->getMessage();
    }

$conn = null;
?>
```

Εισαγωγή πολλαπλών εγγραφών σε MySQL χρησιμοποιώντας mysqli και PDO

Πολλαπλές δηλώσεις SQL πρέπει να εκτελούνται με τη λειτουργία mysqli_multi_query().

Τα ακόλουθα παραδείγματα προσθέτουν τρεις νέες εγγραφές στον πίνακα «MyGuests»:

Παράδειγμα PHP_SQL16.php (mysqli αντικειμενοστραφής)

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

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

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com');";
$sql .= "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Mary', 'Moe', 'mary@example.com');";
```

```
$sql .= "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Julie', 'Dooley', 'julie@example.com')";

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

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

 Σημειώστε ότι κάθε δήλωση SQL πρέπει να χωρίζεται με ελληνικό ερωτηματικό.

Παράδειγμα PHP_SQL17.php (mysqli διαδικαστική)

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

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

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com');";
$sql .= "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Mary', 'Moe', 'mary@example.com');";
$sql .= "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Julie', 'Dooley', 'julie@example.com')";

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

mysqli_close($conn);
?>
```

Προσοχή: ο τρόπος PDO είναι λίγο διαφορετικός:

Παράδειγμα PHP_SQL18.php (PDO)

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

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

   // begin the transaction
   $conn->beginTransaction();
   // our SQL statememtns
   $conn->exec("INSERT INTO MyGuests (firstname, lastname, email)
   VALUES ('John', 'Doe', 'john@example.com')");
   $conn->exec("INSERT INTO MyGuests (firstname, lastname, email)
   VALUES ('Mary', 'Moe', 'mary@example.com')");
   $conn->exec("INSERT INTO MyGuests (firstname, lastname, email)
   VALUES ('Julie', 'Dooley', 'julie@example.com')");

   // commit the transaction
   $conn->commit();
   echo "New records created successfully";
   }
catch(PDOException $e)
   {
   // roll back the transaction if something failed
   $conn->rollback();
   echo "Error: " . $e->getMessage();
   }

$conn = null;
?>
```

PHP προετοιμασμένες (prepared) δηλώσεις

Οι έτοιμες (ή αλλιώς προετοιμασμένες) δηλώσεις είναι πολύ χρήσιμες για αποφυγή τυχόν κακόβουλων SQL injections (μολύνσεων).

Προετοιμασμένες δηλώσεις & Δεσμευτικές Παράμετροι

Μια προετοιμασμένη δήλωση είναι ένα χαρακτηριστικό που χρησιμοποιείται για να εκτελέσει τις ίδιες (ή κάπως παρόμοιες) δηλώσεις SQL, επανειλημμένα με υψηλή απόδοση. Στη βιβλιογραφία η δυνατότητα αυτή αναφέρεται και ως ***Dynamic SQL***.

Οι έτοιμες δηλώσεις λειτουργούν ως εξής:

1. Προετοιμασία: Ένα πρότυπο δήλωσης SQL δημιουργείται και αποστέλλεται στη βάση δεδομένων. Ορισμένες τιμές μένουν απροσδιόριστες, που ονομάζονται παράμετροι (με την ένδειξη "?"). Παράδειγμα: INSERT INTO MyGuests VALUES (?, ?, ?)
2. Η βάση δεδομένων αναλύει, μεταγλωττίζει, βελτιστοποιεί τα ερωτήματα στο πρότυπο δήλωσης SQL, και αποθηκεύει το αποτέλεσμα χωρίς να το εκτελέσει.
3. Εκτέλεση: Σε μεταγενέστερο χρόνο, η εφαρμογή συνδέει τις τιμές με τις παραμέτρους, και η βάση δεδομένων εκτελεί τη δήλωση. Η αίτηση μπορεί να εκτελέσει τη δήλωση όσες φορές θέλει με διαφορετικές τιμές

Σε σύγκριση με την άμεση εκτέλεση των δηλώσεων SQL, οι έτοιμες δηλώσεις έχουν δύο βασικά πλεονεκτήματα:

- Οι έτοιμες δηλώσεις μειώνουν το χρόνο ανάλυσης καθώς η προετοιμασία για το ερώτημα γίνεται μόνο μία φορά (αν και η δήλωση εκτελείται πολλές φορές)
- Οι δεσμευτικές παράμετροι ελαχιστοποιούν το εύρος ζώνης για το διακομιστή, όταν χρειάζεται να στέλνετε κάθε φορά μόνο τις παραμέτρους, και όχι ολόκληρο το ερώτημα
- Οι έτοιμες δηλώσεις είναι πολύ χρήσιμες έναντι των SQL injections, επειδή οι τιμές των παραμέτρων, μεταδίδονται αργότερα χρησιμοποιώντας ένα διαφορετικό πρωτόκολλο. Αν το γνήσιο πρότυπο δήλωσης δεν προέρχεται από εξωτερική είσοδο, δεν μπορεί να συμβεί κάποιο SQL injection.

Προετοιμασμένες Δηλώσεις στην mysqli

Το ακόλουθο παράδειγμα χρησιμοποιεί έτοιμες δηλώσεις και δεσμεύει παραμέτρους στη mysqli:

Παράδειγμα PHP_SQL19.php (mysqli με prepared δηλώσεις)

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

```
$password = "";
$dbname = "myDB";

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

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

// prepare and bind
$stmt = $conn->prepare("INSERT INTO MyGuests (firstname, lastname, email)
VALUES (?, ?, ?)");
$stmt->bind_param("sss", $firstname, $lastname, $email);

// set parameters and execute
$firstname = "John";
$lastname = "Doe";
$email = "john@example.com";
$stmt->execute();

$firstname = "Mary";
$lastname = "Moe";
$email = "mary@example.com";
$stmt->execute();

$firstname = "Julie";
$lastname = "Dooley";
$email = "julie@example.com";
$stmt->execute();

echo "New records created successfully";

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

Γραμμές κώδικα προς εξήγηση από το πιο πάνω παράδειγμα:

"INSERT INTO MyGuests (firstname, lastname, email) VALUES (?, ?, ?)"

Στην SQL, εισάγουμε ένα ερωτηματικό (?) που θέλουμε να αντικαταστήσει ένα ακέραιο (integer), αλφαριθμητικό (string), έναν διπλής ακρίβειας αριθμό (double) ή ένα πεδίο

blob (binary large object – ως τύπος στην SQL χρησιμοποιείται για να βάζουμε μεγάλου όγκου δεδομένα, πχ. εικόνες, video, κλπ.)

Στη συνέχεια, ρίξτε μια ματιά στην λειτουργία bind_param ():

$stmt->bind_param("sss", $firstname, $lastname, $email);

Η λειτουργία αυτή δεσμεύει τις παραμέτρους στο ερώτημα SQL και αφηγείται στην βάση δεδομένων ποιες είναι οι παράμετροι. Η «sss» παράμετρος απαριθμεί τα είδη των δεδομένων που είναι οι παράμετροι. Ο χαρακτήρας s λέει στη MySQL ότι η παράμετρος είναι ένα string.

Η παράμετρος μπορεί να είναι μία από τις τέσσερις κατηγορίες:

- i - integer
- d - double
- s - string
- b - blob

Πρέπει να έχουμε ένα από αυτά για κάθε παράμετρο.

Λέγοντας στη MySQL τι είδους δεδομένα να αναμένει, ελαχιστοποιείται ο κίνδυνος των SQL injections.

Σημείωση: Εάν θέλετε να εισάγετε δεδομένα από εξωτερικές πηγές (όπως η εισαγωγή από το χρήστη), είναι πολύ σημαντικό το γεγονός ότι τα δεδομένα επικυρώνονται.

Προετοιμασμένες - έτοιμες Δηλώσεις PDO

Το ακόλουθο παράδειγμα χρησιμοποιεί έτοιμες δηλώσεις και δεσμεύει παραμέτρους PDO:

Παράδειγμα PHP_SQL20.php (PDO με prepared δηλώσεις)

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

try {
   $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username,
$password);
```

```
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // prepare sql and bind parameters
    $stmt = $conn->prepare("INSERT INTO MyGuests (firstname, lastname, email)
    VALUES (:firstname, :lastname, :email)");
    $stmt->bindParam(':firstname', $firstname);
    $stmt->bindParam(':lastname', $lastname);
    $stmt->bindParam(':email', $email);

    // insert a row
    $firstname = "John";
    $lastname = "Doe";
    $email = "john@example.com";
    $stmt->execute();

    // insert another row
    $firstname = "Mary";
    $lastname = "Moe";
    $email = "mary@example.com";
    $stmt->execute();

    // insert another row
    $firstname = "Julie";
    $lastname = "Dooley";
    $email = "julie@example.com";
    $stmt->execute();

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

PHP Επιλέξτε δεδομένα από MySQL

Η δήλωση SELECT χρησιμοποιείται για την επιλογή των δεδομένων από έναν ή περισσότερους πίνακες:

```
SELECT column_name(s) FROM table_name
```

ή μπορούμε να χρησιμοποιήσουμε τον χαρακτήρα * για να επιλέξουμε όλες τις στήλες από τον πίνακα:

SELECT * FROM table_name

Επιλέξτε δεδομένα με mysqli

Το ακόλουθο παράδειγμα επιλέγει τις στήλες id, όνομα και επώνυμο από τον πίνακα MyGuests και τις εμφανίζει στην σελίδα:

Παράδειγμα PHP_SQL21.php (mysqli αντικειμενοστραφής)

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

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

$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = $conn->query($sql);

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

Γραμμές κώδικα προς εξήγηση από το πιο πάνω παράδειγμα:

Πρώτα, έχουμε δημιουργήσει ένα ερώτημα SQL που επιλέγει τις στήλες id, firstname και lastname από τον πίνακα MyGuests. Η επόμενη γραμμή κώδικα εκτελεί το ερώτημα και βάζει τα δεδομένα που προκύπτουν σε μια μεταβλητή που ονομάζεται $result.

Στη συνέχεια, η συνάρτηση NUM_ROWS () ελέγχει εάν υπάρχουν περισσότερες από μηδέν γραμμές που επιστρέφονται.

Εάν υπάρχουν περισσότερες από μηδέν γραμμές που επιστρέφονται, η συνάρτηση fetch_assoc() βάζει όλα τα αποτελέσματα σε ένα συσχετιζόμενο πίνακα (associative array), τον οποίο μέσω βρόχου (loop) μπορούμε να διατρέξουμε. Ο βρόχος while () επαναλαμβάνεται για το σύνολο των αποτελεσμάτων και εξάγει τα δεδομένα από τις στήλες id, firstname και lastname.

Το ακόλουθο παράδειγμα δείχνει την ίδια λειτουργία με το παραπάνω παράδειγμα, κατά τον mysqli διαδικαστικό τρόπο. Παρατηρήστε ότι σε αυτό το παράδειγμα οι εγγραφές δεν εμφανίζονται με κενή γραμμή ανάμεσά τους γιατί η echo έχει μόνο ένα
:

Παράδειγμα PHP_SQL22.php (mysqli διαδικαστικό)

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

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

$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = mysqli_query($conn, $sql);

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

mysqli_close($conn);
?>
```

Μπορείτε επίσης να βάλετε το αποτέλεσμα σε έναν πίνακα HTML:

Παράδειγμα PHP_SQL23.php (mysqli αντικειμενοστραφής)

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

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

$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = $conn->query($sql);

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

Προσοχή:
Παρατηρήστε πώς αλλάζει η εμφάνιση των στοιχείων του πίνακα, όταν βάλουμε τον ίδιο κώδικα php μέσα σε αρχείο html που ακολουθεί κανόνες μορφοποίησης CSS3 (αρχείο PHP_SQL23b.php):

```
<!DOCTYPE html>
<html>
        <head>
                <style>
                        table, th, td {
                                border: 1px solid black;
                        }
```

```
        </style>
</head>
<body>
        <?php
        …
        ?>

</body>
</html>
```

ID	Name
1	John Doe
5	John Dickens
6	Mary Dickens
7	Julie Dooley
8	John Doe
9	Mary Moe
10	Julie Dooley
11	John Doe
12	Mary Moe
13	Julie Dooley
14	John Doe
15	John Doe

Επιλέξτε δεδομένα με PDO (+ prepared δηλώσεις)

Το ακόλουθο παράδειγμα χρησιμοποιεί προετοιμασμένες δηλώσεις.

Επιλέγει τις στήλες id, όνομα και το επώνυμο από τον πίνακα MyGuests και τiw εμφανίζει σε έναν πίνακα HTML. Οι κανόνες μορφοποίησης εδώ, παράγονται με echo από την php:

Παράδειγμα PHP_SQL24.php (PDO)

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

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

```
    }

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

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

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

$servername = "localhost";
$username = "root";
$password = "";
$dbname = "myDBPDO";

try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username,
$password);
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $stmt = $conn->prepare("SELECT id, firstname, lastname FROM MyGuests");
    $stmt->execute();

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

Id	Firstname	Lastname
1	John	Doe
4	John	Dickens
5	John	Doe
6	Mary	Moe
7	Julie	Dooley
8	John	Doe
9	Mary	Moe
10	Julie	Dooley
11	John	Doe
12	John	Doe
13	Mary	Moe
14	Julie	Dooley

Διαγραφή δεδομένων από έναν πίνακα MySQL χρησιμοποιώντας mysqli και PDO

Η δήλωση DELETE χρησιμοποιείται για τη διαγραφή εγγραφών από έναν πίνακα:

```
DELETE FROM table_name
WHERE some_column = some_value
```

Ανακοίνωση για το WHERE, στο DELETE σύνταξη: Η πρόταση WHERE προσδιορίζει ποια αρχεία πρέπει να διαγραφούν. Εάν παραλείψετε την πρόταση WHERE, όλα τα αρχεία θα διαγραφούν!

Έστω ότι έχουμε τα ακόλουθα δεδομένα στον «MyGuests" πίνακα:

id	firstname	lastname	email	reg_date
1	John	Doe	john@example.com	2014-10-22 14:26:15
2	Mary	Moe	mary@example.com	2014-10-23 10:22:30
3	Julie	Dooley	julie@example.com	2014-10-26 10:48:23

Το ακόλουθο παράδειγμα διαγράφει την εγγραφή με id = 3 στο "MyGuests" πίνακα:

Παράδειγμα PHP_SQL25.php (mysqli αντικειμενοστραφής)

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

```
$username = "root";
$password = "";
$dbname = "myDB";

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

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

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

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

Το ακόλουθο παράδειγμα διαγράφει την εγγραφή με id = 4 στο "MyGuests" πίνακα:

Παράδειγμα PHP_SQL26.php (mysqli διαδικαστική)

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

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

// sql to delete a record
$sql = "DELETE FROM MyGuests WHERE id=4";

if (mysqli_query($conn, $sql)) {
   echo "Record deleted successfully";
```

```
} else {
   echo "Error deleting record: " . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

Το ακόλουθο παράδειγμα διαγράφει την εγγραφή με id = 3 στο "MyGuests" πίνακα:

Παράδειγμα PHP_SQL27.php (PDO)

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

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

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

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

$conn = null;
?>
```

Μετά την καταχώρηση που διαγράφεται, ο πίνακας θα μοιάζει με αυτόν:

id	firstname	lastname	email	reg_date
1	John	Doe	john@example.com	2014-10-22 14:26:15
2	Mary	Moe	mary@example.com	2014-10-23 10:22:30

PHP ενημέρωση δεδομένων σε MySQL

Ενημέρωση δεδομένων σε έναν πίνακα MySQL χρησιμοποιώντας mysqli και PDO

Η δήλωση UPDATE χρησιμοποιείται για την ενημέρωση των υφιστάμενων εγγραφών σε έναν πίνακα:

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

Ανακοίνωση για το WHERE, στη σύνταξη UPDATE: Η πρόταση WHERE προσδιορίζει ποια αρχεία θα πρέπει να ενημερωθούν. Εάν παραλείψετε την πρόταση WHERE, όλα τα αρχεία που θα πρέπει να ενημερωθεί!

Έστω ο «MyGuests" πίνακας ότι περιέχει:

id	firstname	lastname	email	reg_date
1	John	Doe	john@example.com	2014-10-22 14:26:15
2	Mary	Moe	mary@example.com	2014-10-23 10:22:30

Τα ακόλουθα παραδείγματα ενημερώνουν την εγγραφή με id = 2 στο "MyGuests" πίνακα:

Παράδειγμα PHP_SQL28.php (mysqli αντικειμενοστραφής)

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

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

$sql = "UPDATE MyGuests SET lastname='Dickens' WHERE id=2";

if ($conn->query($sql) === TRUE) {
```

```
   echo "Record updated successfully";
} else {
   echo "Error updating record: " . $conn->error;
}

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

Παράδειγμα PHP_SQL29.php (mysqli διαδικαστική)

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

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

$sql = "UPDATE MyGuests SET lastname='Georgiadis' WHERE id=2";

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

mysqli_close($conn);
?>
```

Προσοχή: Με χρήση PDO, μέσω της rowCount() η εικόνα του αποτελέσματος της update είναι πιο ακριβής.

Παράδειγμα PHP_SQL30.php (PDO)

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

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

   $sql = "UPDATE MyGuests SET lastname=Georgiadis WHERE id=2";

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

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

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

$conn = null;
?>
```

Μετά την ενημέρωση της εγγραφής, ο πίνακας θα μοιάζει με αυτό:

id	firstname	lastname	email	reg_date
1	John	Doe	john@example.com	2014-10-22 14:26:15
2	Mary	Georgiadis	mary@example.com	2014-10-23 10:22:30

PHP Επιλογές περιορισμένων δεδομένων από MySQL

Επιλογές περιορισμένων δεδομένων από μια βάση δεδομένων MySQL

Η MySQL παρέχει LIMIT όριο που χρησιμοποιείται για να καθορίσει τον αριθμό των εγγραφών που θα επιστραφούν.

Το LIMIT όριο καθιστά εύκολη την κωδικοποίηση αποτελεσμάτων πολλαπλών σελίδων ή τη σελιδοποίηση με SQL, και είναι πολύ χρήσιμο σε μεγάλους πίνακες. Η επιστροφή ενός μεγάλου αριθμού εγγραφών μπορεί να επηρεάσει την απόδοση.

Ας υποθέσουμε ότι θέλουμε να επιλέξετε όλα τα αρχεία 1 έως 30 (συμπεριλαμβανομένου) από έναν πίνακα που ονομάζεται "Orders". Το ερώτημα SQL τότε θα μοιάζει με αυτό:

$sql = "SELECT * FROM Orders LIMIT 30";

Όταν το παραπάνω ερώτημα SQL τρέξει, θα επιστρέψει τις πρώτες 30 εγγραφές.

Τι γίνεται αν θέλουμε να επιλέξουμε τις εγγραφές 16-25 (συμπεριλαμβανομένου);

Η Mysql παρέχει επίσης έναν τρόπο για να το χειριστεί αυτό: με τη μέθοδο OFFSET.

Το παρακάτω ερώτημα SQL λέει "να επιστρέψει μόνο 10 αρχεία, ξεκίνησε από την εγγραφή 16 (λόγω του OFFSET 15)":

$sql = "SELECT * FROM Orders LIMIT 10 OFFSET 15";

Μπορείτε επίσης να χρησιμοποιήσετε μια μικρότερη σύνταξη για να επιτευχθεί το ίδιο αποτέλεσμα:

$sql = "SELECT * FROM Orders LIMIT 15, 10";

Σημειώστε ότι οι αριθμοί αντιστρέφονται όταν χρησιμοποιείτε το κόμμα.